Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

A Novel Method for Resource Efficient Security Service Chain Embedding Oriented to Cloud Datacenter Networks

Wenxin Qiao, Yicen Liu, Leiping Xi, Xi Li, Zhiwei Li, Donghao Zhao, and Yu Lu
Shijiazhuang Campus, Army Engineering University, Shijiazhuang, 050003, China

Corresponding author: Yicen Liu (e-mail: yicen_sdnfv@ 126.com).

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 62071483. The authors would like to thank the editor and anonymous expert reviewers for the instructive comments, and we appreciate your warm work earnestly.

ABSTRACT An important and promising application for the network function virtualization (NFV) technology is in network security, where can dynamically and flexibly accomplish the chaining virtualized security network functions (VSNFs), e.g., network address translation, antispam and packet filter firewall, etc., and thus inspect, monitor or filter traffic flows in the cloud datacenter networks. However, the traffic flows addressed by the VSNFs mainly depend on the security service requirements from mobile users, such as the network security level, end-to-end latency, and security resource, etc. Considering the dynamic nature of cloud datacenter networks, determining the embedding of VSNFs and routing security service paths that optimizes the security resource utilization is a challenging problem, particularly without violating the end-to-end delay constraints and security service requirements. This can also be called security service chain dynamic embedding problem (SSC-DMP). In this paper, we present an NFV-enabled framework for a system that achieves the SSC dynamic embedding in the cloud datacenter networks. We first formulate an integer linear programming (ILP) model to solve the SSC-DMP exactly in small-scale network topology. Then, in order to reduce the time complexity when applying the large-scale network topology, we propose an efficient SSC dynamic embedding solution that is based on the particle swarm optimization. Extensive simulation results show that the proposed algorithm could significantly outperform the current benchmarks at least 35.2% and 23.1% in terms of resource consumption and end-to-end delay, respectively.

INDEX TERMS Network function virtualization (NFV), security network functions (VSNFs), security service chain (SSC), dynamic embedding, integer linear programming (ILP), particle swarm optimization

I. INTRODUCTION

Typically, network security implemented by the security service chain (SSC) phenomenon arises from the fact that the traffic flows are required to steer through a set of security middleboxes in the predefined specified sequence. This is typically referred to as the static security service chaining [1-3]. It can be observed that this task is performed by manually crafting the routing table entries, which can be recognized to be complicated, slow, wasteful and highly error-prone, and thus it can not adapt to the dynamic and diversity security requirements of future security service applications oriented to the cloud datacenter networks, i.e., the fixed security middlebox placement can not be optimal for all possible traffic patterns in the cloud datacenter network scenarios, which remarkably requires the cumbersome and costly security middlebox upgrades/ relocation [4, 5].

Driven by the diversity of heterogeneous network security services, the network framework design has been transiting from the monolithic pattern to the softwaried paradigm, which is mainly supported by the network function virtualization (NFV) and software-defined network (SDN) technologies [6-9]. These technologies present themselves as revolution pivotal network architectural design concepts that leverage the virtualization and cloud infrastructure elasticity for the purpose of supporting this quantum leap of the existing packet core which, in turn, leads to the remarkable improvement of provisioned security network services. Through the NFV technology [10], security network services are provisioned in software-enabled security network functions or elements by leveraging the virtualization technology, i.e., processing traffic with virtualized security network functions (VSNFs). The SDN framework is used to traverse the traffic flows through the predefined VSNFs at network edge in order based on the orchestration strategy, thus forming the security service chain (SSC).

The SSC technology has been widely regarded as the promising and important application area for future cloud datacenter networks, where can dynamically and flexibly accomplish the chaining of the VSNFs, e.g., network address translation, antispam and packet filter firewall, etc., and thus inspect, monitor, or filter traffic flows. This brings several advantages to the field of cloud datacenter networks that can be summarized [11-14]: (i) highly customizable security services according to the dynamic and different delay requirements of mobile users; (ii) rapid actions to reconstruct the security system when facing new security threats or network attacks; (iii) low Capital Expenditure (CAPEX) and Operating Expenditure (OPEX) for network operators (NOs). Due to the advantage of SSC, it has been widely recognized as an important technology that provides a new way to deal with the network security problem oriented to the cloud datacenter networks.

In the application of the SSC technology for the cloud datacenter networks, the efficient online dynamic SSC resource allocation is one of the most challenging problems in the field of software-defined security [6, 14]. With the rapid increase of mobile users, security service demands from mobile users exponentially grow, that is to say, network security problem has gradually emerged in cloud datacenter networks when tens of thousands of IoT terminals access the network [15]. It is difficult to adopt traditional static service mode that is mainly based on the specialized, closed, proprietary hardware. To deal with the network security problem in the cloud datacenter networks, employing the NFV-enabled SSC makes it possible to allow the arbitrary VSNFs to be placed in any cloud servers through an automatedly and logically centralized VSNF management system [16]. Security service at the cloud datacenter networks requires the different and rigorous performance indexes, such as the security levels, end-to-end latency, and the resource utilization. In this paper, we focus on dealing with the *SSC dynamic embedding problem (SSC-DMP) oriented to cloud datacenter networks*, which has not well explored in the prior related literatures.

While a number of current literatures solve the different requirement-driven SFC resource allocation problem [17-21] in the generic NFV environment, few works have taken the security-driven SFC embedding in the cloud datacenter networks into account, which is actually a fairly recent issue. There are some pioneering studies with regard to the SSC-DMP [6, 13-14]. However, their studies mostly focus on the optimization of SSC resource allocation, which fail to consider the requested security levels of the VSNFs along an SSC. Without taking the specific security level of the VSNFs into account, the current related literatures might easily result in more security resource wastage and lower resource utilization in the optimization. Due to the dynamic nature of cloud datacenter networks, efficiently achieving the SSC resource allocation is a very urgent and promising problem to be solved. In this work, we study three main sub-problems: *How to efficiently utilize the substrate security resources of cloud datacenter by considering the resource constraints, security constraints, end-to-end delay constraints? How to adaptively determine the appropriate VSNFs based on the specific security levels and the security service requirements? How to design an online dynamic novel algorithm that can adapt to the achieve SSC dynamic embedding oriented to cloud datacenter networks?*

In our work, we are motivated by the emerging enormous number of mobile users that significantly can lead to the abundant number of security service demands. Thus, there is need for an automated and optimal way of online dealing with the SSC-DMP, which can dynamically and flexibly guarantee the service security in cloud datacenter network scenarios. Firstly, we introduce an SSC dynamic embedding framework for the cloud datacenter networks and present the network model for SSC. Then, we formulate the SSC-DMP in cloud datacenter networks as integer linear programming (ILP)-based optimization problem, thus exactly solving the SSC-DMP in the small-scale network topology. Then, in order to reduce the time complexity when applying the large-scale network topology, we propose an efficient SSC dynamic embedding solution that is based on the particle swarm optimization (called PSO). Extensive simulation results demonstrate that the proposed PSO algorithm can efficiently deal with the SSC-DMP in the cloud datacenter networks and can significantly outperform three previous algorithms. Specifically, the main contributions of this paper can be summarized as follows.

1) We firstly propose an SSC dynamic embedding framework for the cloud datacenter networks, and present the network model for NFV-enabled security service chaining.
2) We take into account a small-scale practical cloud datacenter network application scenario for the SSC-DMP, and formulate an ILP model to exactly solve the optimization problem in such small-scale network scenarios.
3) In order to reduce the time complexity, we propose an efficient SSC dynamic embedding solution that is based on the particle swarm optimization (PSO) algorithm.
4) The obtained simulation results on the real-world network topologies show that the proposed algorithm could significantly perform better than three previous baseline methods (i.e., Tabu Search and two Greedy biased approaches) in terms of the end-to-end delay, security resource consumption and algorithm run time.

The rest of our manuscript is organized in the following. Section II summarizes and analyzes the most related works. Section III gives an SSC dynamic embedding in the cloud datacenter networks and presents the network model. Section IV formulates the ILP-based SSC-DMP for cloud datacenter networks. Furthermore, our proposed approach for SSC-DMP is presented in Section V. Section VI shows the obtained simulation results and the corresponding analysis. Eventually, we conclude our manuscript.

II. RELATED WORKS

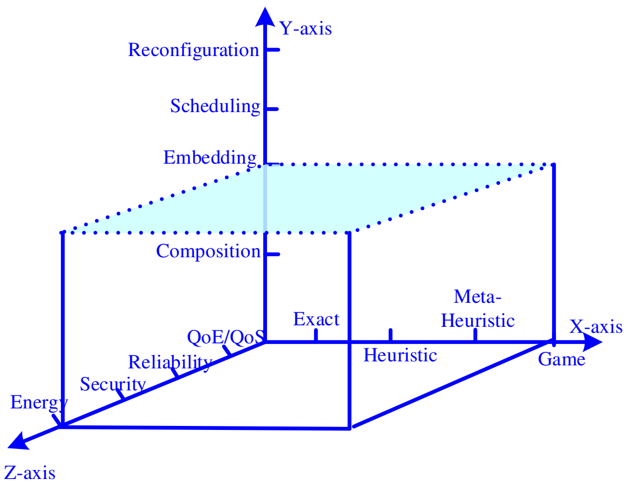


FIGURE 1. Three-dimensional space regarding the SFC-RA

With the rapid growth of the virtualization-based NFV technology, the SFC resource allocation (SFC-RA) problem has been investigated in the literature from various aspects (e.g., strategies, objectives, and stages). In the related work, we summarize and present the three-dimensional space in the regard of the SFC-RA, as shown in Fig. 1. To the best of our knowledge, the current literatures in this regard are all within the three-dimensional space. As for the $\mathcal{X}$-axis (strategy), such kind of the SFC-RA can be addressed by the exact, heuristic, meta-heuristic, game, etc. For the stage of the SFC-RA ($\mathcal{Y}$-axis), the main stages of the SFC-RA include the SFC composition, SFC embedding, SFC scheduling and SFC reconfiguration, and SFC embedding is regarded as one of the most crucial techniques in the field of SFC-RA that is our focus in this paper. Besides, as for the $\mathcal{Z}$-axis (strategy), some classical attributes include the QoS/QoE, security, reliability/availability, and energy, etc. Although a number of related works have studied the SFC embedding in cloud datacenter network scenarios, much less attention has been paid to the SSC-DMP, which is recognized as one of the most pressing problems in the field of network security oriented to the cloud datacenters. From the large angle, we present the related works in terms of the QoS/QoE-driven SFC resource allocation, the availability-driven SFC resource allocation and the security-driven SFC resource allocation, which are summarized as follows:

A. QoS/QoE-DRIVEN SFC RESOURCE ALLOCATION

One can not ignore the significance of QoS/QoE in the network optimization when studying the SFC-RA problem in the cloud datacenter network framework. Service providers should consider the diversity QoS/QoE-driven requirements to satisfy the promised SLAs for requested applications. There are mainly four categories of methods including the exact, heuristic, meta-heuristic and game, etc. In order to shorten the cloud servicing delay using approximate algorithm, Yang *et al.* [15] study the problem of how to place VNFs on edge and public clouds and route the traffic among adjacent VNF pairs, which minimizes the maximum link load ratio and guarantees the delay constraints. The authors creatively propose an exact integer non-linear program (INLP). In order to solve the optimization problem, which remains NP-hard, in polynomial time, the authors propose a randomized rounding approximation method called RRVA to solve this problem. However, the proposed RRAV directly transforms from the INLP to the linear programming (LP), which would easily drop some feasible solutions. Bhamare *et al.* [22] discuss the problem of designing the optimal scheduling of microservices across multiple clouds where microservices are organized as service chains and service function chains should be efficiently scheduled considering the dynamic traffic flows and respecting QoS requirements. The authors propose a fair weighted affinity scheme for SFC scheduling, and evaluate the proposed heuristic approach over various microservice deployment and communication options over clouds, and demonstrate the efficiency of the heuristic compared to the standard greedy scheduling algorithms. Meanwhile, the study in [23] proposes a genetic algorithm (GA)-based strategy to dynamically place VNF forwarding graphs (VNF-FGs) with the objectives of enhancing service performance, reducing VNF deployment cost, and meeting constraints such as CPU, memory, and disk capabilities. Chen *et al.* [24] formulate the SFC outsourcing problem as an ILP model, and propose the dynamic programming-based Viterbi method of outsourcing SFC to multiple edge clouds. However, its formulated model is not a complete model for dynamic and complex multiple domain networks, where the constraints and objectives are changeable.

B. AVAILABILITY-DRIVEN SFC RESOURCE ALLOCATION

The services provided by application service providers (ASPs) as SFC should be always online for the mobile users. The action of the dynamic addition or removal should be taken in a seamless manner. The current literatures regarding the availability-driven SFC dynamic orchestration can be divided into two components including VNF backup and VNF migrations. Determining the best number of VNF backups is one of the most important sub-problems in the field of the availability-driven SFC dynamic orchestration. For example, Qu *et al.* [25] propose an incremental method that can determine the number of required VNF backups, thus achieving the available-aware joint VNF embedding and traffic flows. The authors employ a SFC resource allocation strategy that could significantly trade off the reliability, CPU and bandwidth consumption of a service chain. Fan *et al*. [26] propose an efficient redundancy model that is called a joint protection (JP). Based on the JP redundancy model, the authors further propose a greedy strategy, which can select the two least reliable VNFs for redundancy. However, this might easily make it easy to obtain the local optimal solution. Few literatures gradually start to adopt the reinforcement learning (RL) to the field of availability-driven SFC dynamic orchestration [27-29]. For example, Liu *et al*. [28] formulate the availability/reliability-aware service chaining problem as a mixed integer linear program (MILP) model. To reduce the time complexity, the authors further propose a novel online algorithm that employs the JP redundancy model and Q-Learning-based backup selection. However, we find it hard to apply the approach to the large-scale network scenarios. But beyond that, the VNF migration mechanism can be also employed to improve the availability/reliability of the service chain. For example, Carpio *et al.* [30] propose a linear program (LP)-based model for active-active adjustment according to both VNF backups and VNF migrations, and hence improve the reliability of service chain. The authors also present a N-to-N adjustment mechanism to speed-up the service recovery after VNF-enabled server failures. The approach is proposed based on the implicit scheme, which increase the spare resources to each replica. However, it can only support only one replica per VNF.

C. SECURITY-DRIVEN SFC RESOURCE ALLOCATION

Network security problem is another essential concern for SDN/NFV-enabled SFC framework. In addition, combining SFC and MEC provides a novel solution to guarantee the network security in the edge cloud networks. However, there only have been some studies with respect to the security in the core cloud networks, and the most related literatures focus on different perspectives of SSC-DMP. For example, Shin *et al.* [31] pioneered to present a dynamic security service deployment framework (called FRESCO) that accomplishes the flexible orchestration of security services. The authors also demonstrate the effectivity the novel architecture. Liu *et al.* [32] propose an SDN/NFV-enabled SSC orchestration architecture. Based on the architecture, the authors further present a meta-heuristic-based Tabu search algorithm, which aims to minimize the end-to-end delay and security resource cost. The authors' proposed algorithm can obtain the optimal embedding point of the VSNFs via the iterative manner, however, tabu search is strongly determined by the initial solution of VSNFs and the random initial solution might easily reduce the convergence speed of the meta-heuristic-based Tabu search approach. Sendi *et al.* [33] propose a greedy heuristic based VSNF dynamic orchestration approach, which selects the security function instance with the maximum network utility in an iterative cycle. However, as a heuristic method, this approach can easily be trapped into the local optima when the network scale gradually increases. Doriguzzi-Corin *et al.* [13] focus on which VSNFs should apply for a given security application, the placement of VSNFs and the routing of the traffic flows. Considering the requested security demands of security applications and the service policy of NOs, the authors propose a progressive embedding of security services (PESS). However, the work fails to take the security level into account, which easily causes more security resource fragment, security resource wastage and lower resource utilization. Since security-driven SFC embedding can be regarded as the SSC-DMP, we employ the abbreviation "SSC-DMP" to represent the security-driven SFC-DMP in the following sections.

D. SUMMARY

Summarizing the related works mentioned above, the current popular solutions are mostly based on the heuristic-based methods. However, in the cloud network systems where the constraints and objectives are changeable, these types of approaches are not very appropriate since they generally require a total redesign of the heuristic. Furthermore, with heuristics, they have a rapid convergence at the price of the risk of sticking at a local minimum that can not be effective.

In the dynamic and complex cloud datacenter networks, there is need of designing a novel dynamic online method that can rapidly adapt to the scenarios where the constraints and objectives are changeable. Our proposal is based on the meta-heuristic algorithm, which makes it possible to respond effectively to the SFC resource allocation. Compared to the current related literatures regarding the SSC-DMP, there are three main advantages in this work, which can be summarized as follows: Firstly, we focus on the security service chain embedding problem based on the meta-heuristic based PSO algorithm. Unlike the current literatures based on the heuristics, our proposal can easily integrate new objectives or constraints without reconsidering the solution, which is more appropriate for the dynamic network scenarios; Secondly, we take into account the network security in the cloud datacenter network scenarios, where the network security problem is gradually becoming more and more obvious due to the explosive growth of mobile users; Finally, we consider the specific security level-based mechanism, which significantly avoids resulting in more security resource fragments and the wastage of the substrate security resources.

In this work, we firstly present the network security level-based ILP model. In order to reduce the time complexity when applying the large-scale network topology, we propose an efficient SSC dynamic orchestration solution that is based on the PSO algorithm. To the best of our knowledge, the efficient SSC embedding problem for the security resource optimization with the end-to-end latency constraints in cloud datacenter networks has not been investigated before. The objective of our proposal is to minimize the total resource consumption that simultaneously includes the IT resource consumption of VSNFs and bandwidth consumption of the security service path, while guaranteeing the end-to-end latency of the security service path. Note that we are motivated by the current traditional SFC dynamic embedding approaches, and we propose a novel dynamic online approach considering the characteristics of the SSC for the cloud datacenter networks, e.g., the service level of the VSNFs, dynamic SSC requests and accurate security resource allocation. In addition, our work can be also be adjusted to deal with the dynamic and online SFC embedding in generic NFV networks.

III. NETWORK ARCHITECTURE AND NETWORK MODEL

In this section, we introduce the SSC dynamic embedding architecture oriented to cloud datacenter networks. Then, we present the network model for SSC dynamic embedding.

A. NETWORK ARCHITECTURE

Firstly, we present a security system framework according to the ISO7498-2 standard [34], as shown in Fig. 2. The framework consists of five main security services, which includes the anti-repudiation, identity authentication integrity, encryption, and authorization. Each security service consists of one or more security functions (SFs). For example, the security service of encryption contains the security functions of the routing control, traffic analysis and encipherment.

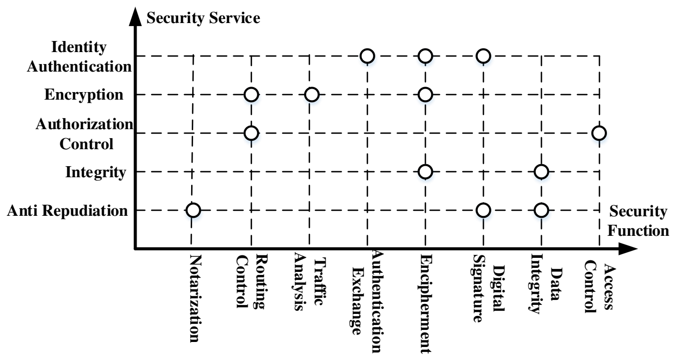


FIGURE 2. **Overview of an open information security framework**

Through the NFV technology, security network services can be provisioned in software-enabled network functions or elements, which can dynamically satisfy different security service demands. Inspired by the literature [4], an SSC dynamic embedding architecture in the cloud datacenter networks is presented, as shown in Fig. 3 In the architecture, the SSC dynamic embedding architecture can be classified into four planes, including the application plane, control plane, virtualization plane and infrastructure plane. At the application plane, different security service requests (SSRs) are generated according to the requirements of mobile users. The control plane receives the SSRs from mobile users. Each SSC includes the VNFs and the corresponding virtual links, which shares the same substrate infrastructures and can be elements, which can dynamically satisfy different security service demands. An SSC dynamic embedding architecture in the cloud datacenter networks is presented, as shown in Fig. 3. In the architecture, the SSC dynamic embedding architecture can be classified into four planes, including the application plane, control plane, virtualization plane and infrastructure plane. At the application plane, different security service requests (SSRs) are generated according to the requirements of mobile users. The control plane receives the SSRs from mobile users. Each SSC includes the VNFs and the corresponding virtual links, which shares the same substrate infrastructures and can be embedded onto the substrate networks via the control plane. NOs can run the SSC embedding algorithms on the control plane to flexibly manage the substrate virtualized resources, thus achieving the VSNFs placement and routing the traffic flows among these adjacent VSNF pairs in the hybrid clouds. SSC consists of a set of heterogenous VSNF Instances (VSNFIs), which can process massive traffic flows for the security applications. Multiple virtualized substrate resources (e.g., CPU, storage, bandwidth, etc.) and the virtualized machines (VMs) are contained at the virtualization plane. The virtualized substrate resources can be dynamically allocated, and thus form a number of independent network slices. The network slicing technology [35] is regarded as an essential enabler that could enable multiple network slices to share the same infrastructure, and a network slice achieves a service logic as SSC that contains the required VSNFs provided by the network. Similar to the literature [36], we can represent an

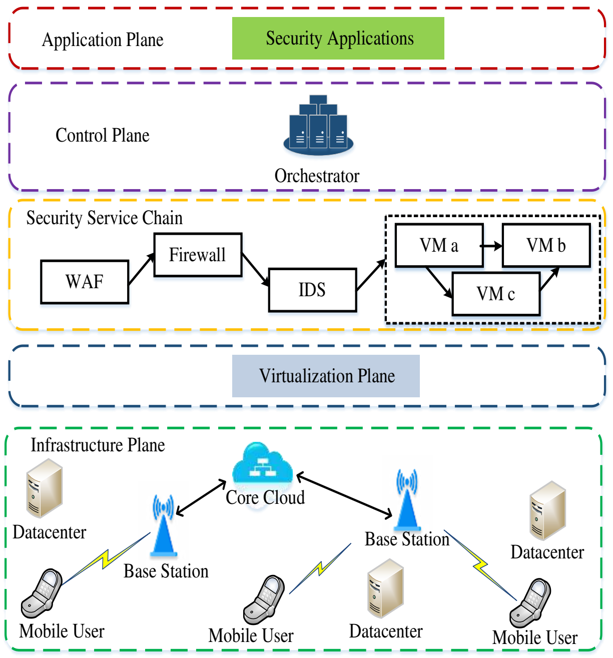


FIGURE 3. **SSC embedding architecture in cloud datacenter networks**

SSC as VSNF-Forwarding Graph (VSNF-FG). An example with four VSNFs in a VSNF-FG is shown in Fig. 3, an SSC consists of Web Application Firewall (WAF), Firewall, Intrusion Detection System (IDS) and VM. Subsets of the VSNFIs comprise a VSNF, for example, VM-a, VM-b and VM-c comprise the security function of the VM. Eventually, the infrastructure plane mainly consists of mobile users and some cloud servers (edge servers and core cloud servers), which are used to support the SSC embedding. At the infrastructure plane, there are the computing resources, storage and network facilities that can process, storage and connect for the security service paths. To better provide the security applications with the network services, cloud servers should be collaborated with each other to achieve SSC embedding according to different security requirements in the cloud datacenter networks.

B. NETWORK MODEL

1) THE NETWORK GRAPH MODEL

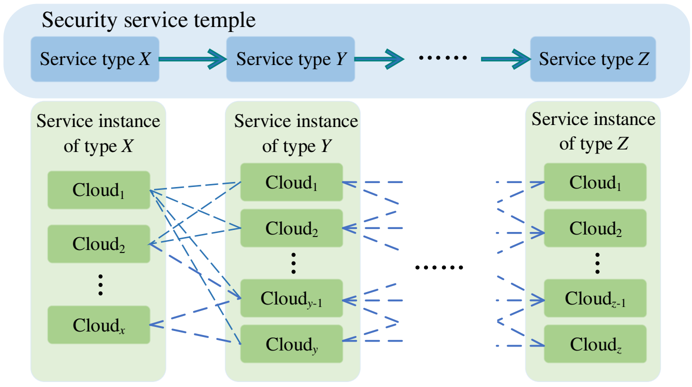


FIGURE 4. SSC request and its corresponding candidate graph

The substrate network and the SSC request can be modelled as an undirected graph and a directed graph, respectively. From the perspective of the control plane, the global view of the substrate networks can be constructed according to the security service request and the substrate network information at the same time. Since the totally ordered SSC is the special format of the partially ordered SSC, we only consider the totally ordered SSC in the model. As shown in Fig. 4, we present the global view of the substrate networks based on the totally ordered SSC. The candidate graph of SSC request would be constructed when receiving an SSC request. In the directed graph, each security service in the SSC request consists of several VSNFIs, which are varying from the embedding locations, the security levels, the processing time, and the remaining resources. Besides, the logical links between the VSNFIs are embedded on the substrate links, which are determined by bandwidth and the transmission delay. The detailed abstract expressions of the substrate network and the SSC request can be described as follows:

Substrate Network: Let $\mathbb{G} = (\mathbb{N}^s, \mathbb{E}^s, A_n, A_e)$ represent the substrate networks with the virtualized security resources, where $\mathbb{N}^s$ denotes the set of the substrate nodes with the IT capacities, $\mathbb{E}^s$ is the set of the substrate links. A_n is the attributes of the node n in the set $\mathbb{N}^s$, i.e., $A_n = \left\{C_n^r, L_n, T_{n,f}^{process}\right\}$, where C_n^r represents the total amount of IT resources, $r \in \mathbb{R} = \{\text{CPU, RAM, HDD}\}$ denotes CPU, memory, and storage resources, respectively. $T_{n,f}^{process}$ denotes the processing time of VSNF $f \in F$ on the substrate node n. L_n is the location of the substrate node n. $A_p = \left\{B_p, T_p^{transfer}\right\}$ is the attributes of the security service path p in the $\mathbb{P}$, where $p \in \mathbb{P}$ represents the substate path on which security service chain is routed along, B_p represents the remaining bandwidth on the link $p \in \mathbb{P}$ and $T_p^{transfer}$ denotes the transmission delay on the security path $p \in \mathbb{P}$. As for the $p \in \mathbb{P}$, it comprises the ordered sequences of the substrate links for security service chain.

Security Service Chain (SSC) Request: Assume that there are different mobile users connecting to the clouds whenever and wherever. Each mobile user asks for different SSC demands, which consists of a predefined ordered VSNFs. Let the directed graph $\mathbb{G}^v = (\mathbb{N}^v, \mathbb{E}^v)$ represent the SSC request, where $\mathbb{N}^v$ and $\mathbb{E}^v$ represent the set of VSNFs and the virtual links, respectively. We also let $r_k=(s_k, d_k, F_k, B^k, C^k, D^k)$ denote the k^{th} SSC request, where s_k and d_k represent the ingress and egress of SSC request, respectively, F_k denotes the VSNF sequence of the k^{th} SSC request (e.g., WAF $\rightarrow$ Firewall $\rightarrow$ IDS), $B^k(e_v)$ is the requested bandwidth resources for the virtual links $e_v \in \mathbb{E}^v$ along the SSC request, $C^k(f)$ is the requested IT resources for the VSNF $f \in \mathbb{F}$ along the SSC request, D^k represents the maximum tolerable end-to-end delay for the k^{th} SSC request.

2) THREE-LEVEL SSC MODEL

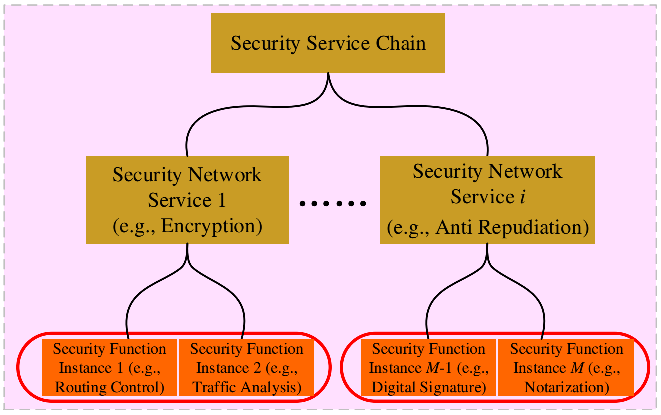

Security Service Chain
Security Network Service 1 (e.g., Encryption)
Security Network Service i (e.g., Anti Repudiation)
Security Function Instance 1 (e.g., Routing Control)
Security Function Instance 2 (e.g., Traffic Analysis)
Security Function Instance M-1 (e.g., Digital Signature)
Security Function Instance M (e.g., Notarization)

FIGURE 5. Overview of the three-level model

In the subsection, the process of SSC can be abstractly described as a three-level embedding model, as shown in Fig. 5. There are two embedding processes in the three-level model, which is from the “Security Service Chain (SSC)” to “Security Network Service (SNS)” and from “Security Network Service (SNS)” to “Security Function Instance (SFI)”. The details of these definitions are as follows:

Definition 1 Security Function Instance (SFI). The SFI represents the basic unit, which can filter and compress the traffic flows from mobile users. In general, the subsets of

SFIs make up a VSNF. For example, the security function of encryption has several SFIs including the advanced encryption standard, the data encryption standard, and the asymmetric encryption standard, etc. Let a vector $\overrightarrow{instance_i}$ represent the basic unit in one security network function. This can be mathematically expressed as

$$\overrightarrow{instance_i}=(0...0\underset{i^{th}}{1}0...0)^T,\ i=1,\ 2,...,n \tag{1}$$

Where n denotes the number of SFIs in one security network function. According to our previous work [6], we consider the VSNFs with different security levels in this paper. Each SFI is regarded as the product of a security level parameter se_i and the vector of security instance unit $\overrightarrow{instance}$. Note that the network security levels are divided into six levels in the information network [37], i.e., $se_i=\{1,\ 2,3,4,5,6\}$. The vector $\overrightarrow{SFI_i}$ can be defined as follows:

$$\overrightarrow{SFI_i}=se_i\overrightarrow{instance_i},\ se_i=\{1,\ 2,3,4,5,6\},\ i=1,\ 2,...,n \tag{2}$$

Definition 2 Security Network Service (SNS). An SNS consists of different SFIs. For example, the SFIs of the routing control, traffic analysis and encipherment comprise the SNS of encryption. Here, the j^{th} $\overrightarrow{\mathrm{SNS}_j}$ can be described as

$$\overrightarrow{SNS_j}=\left\{\overrightarrow{SFI_k},\overrightarrow{SFI_l},...\right\}=\left\{se_k\overrightarrow{instance_k},se_l\overrightarrow{instance_l},...\right\},$$
$$k,\ l=1,\ 2,...,n,\quad j=1,\ 2,...,m,\ se_k,se_l=\{1,\ 2,3,4,5,6\} \tag{3}$$

Definition 3 Security Service Chain (SSC). The SSC is a set of VSNFs, which requires the traffic flows to traverse through the predefined ordered SFIs. Each SSC consists of M VSNFIs. Therefore, the $\overrightarrow{\mathrm{SSC}}$ can be characterized as

$$\overrightarrow{SSC}=\left(\overrightarrow{SFI_1},\overrightarrow{SFI_2},...,\overrightarrow{SFI_M}\right)^T=$$
$$\left(se_1\overrightarrow{instance_1},se_2\overrightarrow{instance_2},...,se_M\overrightarrow{instance_M}\right)^T \tag{4}$$

IV. PROBLEM DESCRIPTION AND COMPLEXITY ANALYSIS

In this section, we first present the problem description of SSC embedding in cloud datacenter networks. Next, we provide an exact formulation for the problem. Finally, we simply analyze the problem complexity.

A. PROBLEM DESCRIPTION

In this paper, we assume that the clouds consist of multiple cloud servers, respectively. The cloud servers can host all SSCs where NOs are serving a batch of SSC requests $\mathbb{R}$. Note that we consider a cloud datacenter case where the VSNFs can be orchestrated in sequential or branching manners [38]. Here, we divide the SSC into two categories, i.e., the totally ordered SSC set and the partially ordered SSC set. In particular, when a mobile user requests a partially ordered SSC, we can take the maximum value of the totally ordered sub-SFCs as the final value. For the substrate network, the substrate networks have a limited capacity. VSNFs can not only be shared by different SSC requests, but also can be deployed on different substrate nodes. Besides, SSC requests contain the specific security level requirements. Based on these considerations, the SSC Dynamic eMbedding Problem (SSC-DMP) in the cloud datacenter networks can be described as follows:

Definition 4. SSC-DMP. Assume that there is an undirected graph $\mathbb{G}=(\mathbb{N}^s,\mathbb{E}^s,A_n,A_e)$, and the kth SSC request $r_k=(s_k,\ d_k,\ F_k,\ B^k,\ C^k,\ D^k)$. For the kth SSC request, SSC-DMP is to focus on the mechanism for accomplishing the VSNFs placement onto the cloud servers, and sending the control command to steer the traffic flows through the predefined VSNFs in order before reaching the egress, such that the total substrate resource consumption is minimized while satisfying the security level, delay, security resource, order and embedding constraints at the same time.

B. MATHEMATICAL FORMULATION OF THE PROBLEM

In this subsection, we formulate the SSC-DMP in the cloud datacenter networks as the integer linear programming (ILP) formulation and the objective is to minimize the total security resource consumption, which can be classified into the VNSF embedding consumption and the virtual link embedding consumption. Here, we begin with some necessary notations and variables.

ILP notations:

$H_{e_v,p}^{n_i,n_j}$: A given Boolean array. It is 1 if the virtual link $e_v\in\mathbb{E}_v$ is traversed by the path $p\in\mathbb{P}$ between on the edge network device $n_i\in\mathbb{N}^s$ and $n_j\in\mathbb{N}^s$; and 0 otherwise.

$\mathbb{R}$, $\mathbb{F}$, $\mathbb{P}$: The set of SSC, VSNFs and paths.

f_k : The flow rate of the kth security service chain $r_k\in\mathbb{R}$.

se_x : The security levels of the VSNFI $v_x\in\mathbb{F}$.

$C(v_x^{se_x})$: The amount of the requested IT resources of a VSNFI $v_x\in\mathbb{F}$ per security level.

C_{n_i} : The remaining available IT resources on the substrate node $n_i\in\mathbb{N}^s$.

$B(e_v)$: The amount of requested bandwidth between the virtual link $e_v\in\mathbb{E}_v$.

B_{e_s} : The available bandwidth on the substrate link $e_s\in\mathbb{E}_s$.

$T_p^{transfer}$: The transmission delay on the security path $p\in\mathbb{P}$.

$T_{n_i,v_x^{se_x}}^{process}$: The processing time of the VSNF $v_x\in\mathbb{F}$ on the substrate node $n_i\in\mathbb{N}^s$.

ILP variables:

$X_{n_i}^{r_i,v_x^{se_x}}$: A given Boolean variable. It is 1 if a request r_i's requested VSNF $v_x\in\mathbb{F}$ is embedded on the substrate node $n_i\in\mathbb{N}^s$; and 0 otherwise.

$Y_{n_i,n_j,p}^{r,v_x^{se_x},v_y^{se_y}}$: A given Boolean variable. It is 1 if a request r_i's requested VSNFs $v_x\in\mathbb{F}$ and $v_y\in\mathbb{F}$ are embedded on the substrate node $n_i\in\mathbb{N}^s$ and $n_j\in\mathbb{N}^s$, respectively, and the path $p\in\mathbb{P}$ is selected; and 0 otherwise.

Objective:

$$\text{Min} \quad Obj = \sum_{r_k \in \mathbb{R}} \sum_{v_x^{se_x} \in \mathbb{F}} \sum_{n_i \in \mathbb{N}^s} se_x \cdot X_{n_i}^{r_i, v_x^{se_x}} \cdot C(v_x^{se_x}) + \sum_{p \in \mathbb{P}} \sum_{r_k \in \mathbb{R}} \sum_{v_x^{se_x}, v_y^{se_y} \in \mathbb{F}} \sum_{n_i, n_j \in \mathbb{N}^s} Y_{n_i, n_j, p}^{r_i, v_x^{se_x}, v_y^{se_y}} \cdot H_{e_v, p}^{n_i, n_j} \cdot f_k \tag{5}$$

Placement Constraints:

$$\sum_{n_i \in \mathbb{N}^s} X_{n_i}^{r_k, v_x^{se_x}} = 1, \ \forall r \in \mathbb{R}, v_x^{se_x} \in \mathbb{F}, se_x = \{1, \ 2, 3, 4, 5, 6\} \tag{6}$$

$$\sum_{p \in \mathbb{P}} \sum_{n_i, n_j \in \mathbb{N}^s} Y_{n_i, n_j, p}^{r_i, v_x^{se_x}, v_y^{se_y}} = 1, \ \forall r_k \in \mathbb{R}, \ v_x^{se_x}, v_y^{se_y} \in \mathbb{F} \tag{7}$$

Path Selection Constraints:

$$X_{n_i}^{r_k, v_x^{se_x}} \cdot X_{n_j}^{r_k, v_y^{se_y}} = \sum_{p \in \mathbb{P}} Y_{n_i, n_j, p}^{r_k, v_x^{se_x}, v_y^{se_y}}, \quad \forall n_i, n_j \in \mathbb{N}^s, \ r_k \in \mathbb{R}, \ v_x^{se_x}, v_y^{se_y} \in \mathbb{F}, \ se_x = \{1, \ 2, 3, 4, 5, 6\} \tag{8}$$

Delay Constraints:

$$\sum_{v_x^{se_x}, v_y^{se_y} \in \mathbb{F}} \sum_{n_i, n_j \in \mathbb{N}^s} \left(X_{n_i}^{r_i, v_x^{se_x}} \cdot X_{n_j}^{r_i, v_y^{se_y}} \cdot \sum_{p \in \mathbb{P}} Y_{n_i, n_j, p}^{r_k, v_x^{se_x}, v_y^{se_y}} \left(T_p^{transfer} + \frac{T_{n_i, v_x^{se_x}}^{process} + T_{n_j, v_y^{se_y}}^{process}}{2} \right) \right) \le D_k, \quad \forall r_k \in \mathbb{R}, \ se_x = \{1, \ 2, 3, 4, 5, 6\} \tag{9}$$

Node Capacity Constraints:

$$\sum_{v_x^{se_x} \in \mathbb{F}} se_x \cdot X_{n_i}^{r_k, v_x^{se_x}} \cdot C(v_x^{se_x}) \le C_{n_i}, \quad \forall n_i \in \mathbb{N}^s, r_k \in \mathbb{R}, \ se_x = \{1, \ 2, 3, 4, 5, 6\} \tag{10}$$

Link Capacity Constraints:

$$\sum_{r \in \mathbb{R}} \sum_{e_s \in \mathbb{E}_s} \sum_{e_v \in \mathbb{E}_v} \sum_{v_x^{se_x}, v_y^{se_y} \in \mathbb{F}} \sum_{n_i, n_j \in \mathbb{N}^s} Y_{n_i, n_j, p}^{r_k, v_x^{se_x}, v_y^{se_y}} \cdot H_{e_v, p}^{n_i, n_j} \cdot f_k \cdot B(e_v) \le B_{e_s}, \quad \forall p \in \mathbb{P}, \ se_x = \{1, \ 2, 3, 4, 5, 6\} \tag{11}$$

Security Level Capacity Constraints:

$$\sum_{n_i \in \mathbb{N}^s} se_x X_{n_i}^{r_k, v_x^{se_x}} \ge SE_x, \quad \forall \ r_k \in \mathbb{R}, \ se_x, SE_x = \{1, \ 2, 3, 4, 5, 6\} \tag{12}$$

Eq. (5) jointly minimizes the total security resource consumption on the cloud servers and the substrate links. The IT resources consumption at the cloud server $n_i \in \mathbb{N}^s$ can be represented as $\sum_{r_k \in \mathbb{R}, v_x^{se_x} \in \mathbb{F}, n_i \in \mathbb{N}^s} se_x \cdot X_{n_i}^{r_k, v_x^{se_x}} \cdot C(v_x^{se_x})$, and the total bandwidth consumption of the substrate link $e_v \in \mathbb{E}_v$ is $\sum_{p \in \mathbb{P}, r_k \in \mathbb{R}, n_i, n_j \in \mathbb{N}^s, v_x^{se_x}, v_y^{se_y} \in \mathbb{F}} Y_{n_i, n_j}^{r_k, v_x^{se_x}, v_y^{se_y}} \cdot H_{e_v, p}^{n_i, n_j} \cdot f_k$. Eq. (6) and Eq. (7) respectively ensure that the VSNFs and the corresponding virtual links along the SSC can choose only one cloud server node and only one sub-chain on the substrate links. Eq. (8) establishes the relationship between the two ILP variables, i.e., $X_{n_i}^{r_k, v_x^{se1}}$ and $Y_{n_i, n_j}^{r_k, v_x^{se_x}, v_y^{se_y}}$,which ensures that the traffic flows traverse the VSNFs in the predefined order. Specifically, when an SSC request r_k places the VSNFs $v_x^{se_x} \in \mathbb{F}$ and $v_y^{se_y} \in \mathbb{F}$ to the cloud server $n_i \in \mathbb{N}^s$ and $n_j \in \mathbb{N}^s$, only one path $p \in \mathbb{P}$ can be selected to use. Eq. (9) guarantee that for each SSC request $r \in \mathbb{R}$, the total response time of each totally ordered sub-SFC should be less than *D*. Note that the sum of $(T_{n_i, v_x^{se_x}}^{process} + T_{n_j, v_y^{se_y}}^{process})/2$ denotes the entire of the VSNF processing time. Since the VSNFs $v_x^{se_x}, v_y^{se_y}, \ldots$ are counted twice in the calculation process, we take the sum of the VSNF processing time and let it divided by two. Eq. (10) ensures that the requested amount of the VSNF $v_x^{se_x} \in \mathbb{F}$ along the SSC should be less than or equal to its security IT resource capacity C_{n_i} of each cloud server $n_i \in \mathbb{N}^s$. Eq. (11) ensures that the total bandwidth consumption $B(e_v)$ across the SSC does not exceed the bandwidth capacity of the substrate link B_{e_s} on the security service path. Eq. (12) guarantees that the security level of the selected VSNFIs should be equal or more than the requested security level of VSNF along the SSC. Note that we select the best match of the security level from the VSNFI set of the security type, and thus save more substrate security resources and improve the security resource utilization.

C. COMPLEXITY ANALYSIS

Based on the formulated SSC-DMP based ILP model, it can be observed that the time complexity of the formulated ILP model is very high because of its Boolean variables and nonlinear constraints, it can not scale well especially with the increase of the network scale. As $|\mathbb{F}| << |\mathbb{N}^s|$, the time complexity the formulated ILP model can be expressed as $\mathcal{O}_{\text{ILP}} = \mathcal{O}(2^{|\mathbb{N}^s|^4})$, which is still so high. Since the time complexity of the ILP is very high, the application of the formulated optimization might be restricted to the small data sets. In addition, a number of literatures [39, 40] have proved that SFC-DMP is NP-Hard, therefore, SSC-DMP is also $\mathcal{NP}$-Hard. Meta-heuristic-based approach [3, 4] has been proved that it can make it possible to respond effectively to the dynamic and online network problem, therefore, we exploit a meta-heuristic-based approach to deal with the dynamic and online SSC-DMP in cloud datacenter networks in the next sub section.

V. META-HEURISTIC EFFICIENT SOLUTION

Considering the exact ILP solution has exponential running time because of its Boolean variables and nonlinear constraints, the exact approach cannot scale well especially when the problem size increases. It is well known that the traditional SFC dynamic embedding problem is regarded as two-fold problem, the first part consists of embedding the VNFs in the cloud servers while the second part consists of steering the traffic flows through the predefined VNFs in order [41, 42]. In this work, we design a novel meta-heuristic-based approach for the dynamic and online SSC embedding problem (called *Particle Swarm Optimization algorithm, PSO*). The proposed PSO approach takes into account the security level requirements of the VSNFs while embedding the VSNFs in clouds based on the traditional SFC dynamic embedding problem, which can be divided into three parts, including the selection of the security service path, the determination of the security level and the embedding of the VSNFs. More specifically, in the first part, for the k^{th} SSC request $r_k=(s_k, d_k, F_k, B^k, C^k, D^k)$, all the feasible paths can be found from the ingress node s_k and the destination node d_k. The shortest paths are calculated by using the *k*-Dijkstra algorithm [3], which minimizes the $Path_{obj} = \omega_1 (T_p^{transfer} / D) + \omega_2 (B(e_v)/B_{e_s})$, where ω_1 and ω_2 are weighting factors that could be used to adjust the relative importance of the delay and bandwidth component. Then, the obtained security service paths can be stored into the set denoted by *Path_k []*. After finishing constructing the set *Path_k []*, we consider the security level constraints, and construct the feasible search space along the $p \in Path_k[]$ denoted by Ω_r *[]*. The set Ω_r *[]* represents the VSNFI set that satisfies the specific security level of the requested VSNFs. Finally, in order to efficiently address the VSNF embedding problem, we assign the VSNFs along the service path $p \in Path_k[]$ by using the PSO algorithm, which minimizes the total security resource consumption. This can ensure our solution obtained by the PSO algorithm for a single security chain is near-optimal. In the following, the entire procedure of the proposed PSO algorithm can be discussed in detail as follows.

A. SECURITY PATH SELECTION PROCEDURE

As for the subproblem of the security path selection, the substrate network $\mathbb{G} = (\mathbb{N}^s, \mathbb{E}^s, A_n, A_e)$ and the SSC request $\mathbb{G}^v = (\mathbb{N}^v, \mathbb{E}^v)$ are regarded as input. The shortest path algorithm called *k*-Dijkstra algorithm is adopted to obtain the feasible paths *Path_k []*, which can be regarded as the output. More specifically, we firstly initialize the shortest path matrix *S*, weighted value matrix *E* and the stack $s_p \neq \varnothing$ (***Lines* 3-4**). In the weighted value matrix *E*, the metrics of the link delay and the link bandwidth are both concluded. Then, we select the ingress node $n_i \in \mathbb{N}$ to check whether the SSC request $r_k \in \mathbb{R}$ satisfies the bandwidth constraints. If the requested VSNF $v_x \in \mathbb{F}$ and the corresponding virtual links $e_v \in \mathbb{E}_v$ can satisfy the substrate resource constraints, the ratio of the requested amount of the delay $T_p^{transfer}$ and the maximum tolerable amount of delay *D*, and the ratio of the requested amount of the bandwidth $B(e_v)$ and the maximum amount of the remaining bandwidth B_{e_s}, are calculated, respectively (***Lines* 5-9**). After that, we use the *k*-Dijkstra algorithm to calculate the shortest paths under the constraints of the bandwidth of the links and the end-to-end latency of the security service chain (***Lines* 10-14**). The whole procedure of the *security path selection* is as shown in **Algorithm 1**.

Algorithm 1 *k*-Dijkstra algorithm for feasible path solutions

```
1: Input: Substrate network graph $\mathbb{G} = (\mathbb{N}^s, \mathbb{E}^s, A_n, A_e)$, SSC request $\mathbb{G}^v = (\mathbb{N}^v, \mathbb{E}^v)$;
2: Output: Path_k [];
3: Initialize the shortest path matrix S and weighted value matrix E;
4: Initialize the stack $s_p \neq \varnothing$ to store the node of the path $p \in \mathbb{P}$;
5: foreach request $r_k \in \mathbb{R}$ do
6:    foreach server $n_i \in s_k$ do
7:       if $IsResourceAvailable(i, j, \mathbb{R})$ then
9:          Calculate the ratios of resource demands and the maximum
            remaining resource, i.e., $T_{i,j}^{transfer}/D$ and $B(e_v)/B_{e_s}$;
10:         function Selection_Shortestpath $(u, d_k, T_{i,j}^{transfer}, B(e_v))$
11:            Use the k-Dijkstra algorithm with minimizing the
               $Path_{obj} = \omega_1 (T_p^{transfer}/D) + \omega_2 (B(e_v)/B_{e_s})$;
12:            Put the current server node u into the stack $s_p$;
13:            if the server node u == egress node d_k then
14:               Put the current security path p into Path_k [];
15:         update $ResourceAvailable(i, j, \mathbb{R})$
16: return S, E, Path_k []
```

B. SECURITY LEVEL DETERMINATION PROCEDURE

The aim of the subproblem is to select the type of the VSNFs and their corresponding security levels. To reduce the search space and improve the efficiency of the proposed algorithm, the VSNFI set S_r for $r \in \mathbb{R}$ is formed, where the $\boldsymbol{S_r}$ is the matrix that satisfies the security level requirements for the $r \in \mathbb{R}$. Let $\overrightarrow{SFI_{ij}} = se_{ij}\overrightarrow{instance_j}$, $se_{ij} = \{1, 2, 3, 4, 5, 6\}$ represent the j^{th} VSNFI on the server node *i* with the security level se_{ij}, and $\overrightarrow{Instance_j} = (0...0\underset{j^{th}}{1}0...0)^T$ denotes the unit instance of the j^{th} VSNFs in a vector format. Therefore, S_r can be mathematically calculated as follows:

$$S_r = R_r - E_r = \begin{bmatrix} SE_{1,1} & SE_{1,2} & \cdots & SE_{1,N} \\ SE_{2,1} & SE_{2,2} & \cdots & SE_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ SE_{M,1} & SE_{M,2} & \cdots & SE_{M,N} \end{bmatrix} \cdot \begin{bmatrix} \overrightarrow{VSNF_1} \\ \overrightarrow{VSNF_2} \\ \vdots \\ \overrightarrow{VSNF_N} \end{bmatrix} - \begin{bmatrix} se_{1,1} & se_{1,2} & \cdots & se_{1,N} \\ se_{2,1} & se_{2,2} & \cdots & se_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ se_{M,1} & se_{M,2} & \cdots & se_{M,N} \end{bmatrix} \cdot \begin{bmatrix} \overrightarrow{instance_1} \\ \overrightarrow{instance_2} \\ \vdots \\ \overrightarrow{instance_N} \end{bmatrix} \tag{13}$$

where $R_r = \left(\sum_{i=1}^{N} SE_{1,i} \overrightarrow{VSNF_i}, \sum_{i=1}^{N} SE_{2,i} \overrightarrow{VSNF_i}, ..., \sum_{i=1}^{N} SE_{M,i} \overrightarrow{VSNF_i} \right)^T$ denotes the security service requirements for the $r \in \mathbb{R}$, and SE_{ij} and $\overrightarrow{VSNF_j} = (0...0 \underset{j^{th}}{1} 0...0)^T$ denote the security level requirement and the requested VSNF along the SSC, respectively. E_r represents the VSNFI set. Here, S_r can be obtained by subtracting the R_r and the E_r, thus obtaining the VSNFI set that meets the security level requirements. Assume that $(S_r)_{ij} \geq 0$, and then we can obtain the feasible VSNFI set Ω_r for the SSC request $r \in \mathbb{R}$, which can be mathematically expressed as

$$\Omega_r = \left\{ instance_j \middle| \begin{array}{c} se_{ij} \mapsto instance_j, se_{ij}, SE_{ij} \in \{1, 2, 3, 4, 5, 6\}, se_{ij}, SE_{ij} \geq 0 \\ n_i \in \mathbb{N}^s, i \in [1, M], j \in [1, N] \end{array} \right\} \tag{14}$$

where $\mapsto$ represents the mapping relationship between the j^{th} VSNFI and its corresponding security level.

As for the subproblem of the security level determination, the feasible path set *Path$_k$[]*, the SSC request $\mathbb{G}^v = (\mathbb{N}^v, \mathbb{E}^v)$ and the substrate network $\mathbb{G} = (\mathbb{N}^s, \mathbb{E}^s, A_n, A_e)$ are regarded as input, then we design the security level selection algorithm (SLSA) to obtain the VSNFI set *Ω_r []*. Specifically, the VSNF selection matrix *V*, security level matrix *L* and the stack $s_l \neq \varnothing$ are firstly initialized (***Lines* 3-4**). Then, we compute the $S_r = R_r - E_r$ to obtain the VSNFI set *Ω_r []* that meets the security level requirements (***Lines* 9-13**). The whole procedure of the security level determination can be depicted in **Algorithm 2**.

Algorithm 2 Security Level Selection Algorithm (SLSA) for the VSNF instance set that satisfies the specific security level requirements

```
1: Input: Substrate network graph $\mathbb{G} = (\mathbb{N}^s, \mathbb{E}^s, A_n, A_e)$, SSC request
   $\mathbb{G}^v = (\mathbb{N}^v, \mathbb{E}^v)$, shortest path set Path_k [];
2: Output: $\Omega_r$ [];
3: Initialize the VSNF selection matrix V and security level matrix L;
4: Initialize the stack $s_l \neq \varnothing$ to store the VSNF instance $v_x \in \mathbb{F}$;
6:    foreach path $p \in Path_k []$ do
7:       Fetch the specific security requirements $R_r$ and the substrate
         remaining security resource $E_r$ for the $r \in \mathbb{R}$;
8:       function Selection_Securitylevel $(VSNF_1, VSNF_N, SE_{i,j}, se_{i,j})$
9:          $S_r = R_r - E_r$;
10:         if $(S_r)_{ij} \geq 0$ then
11:            Push the current $v_x \in \mathbb{F}$ into the stack $s_l$;
12:            if the VSNF f == last VSNF then
13:               Push the current VSNF instances $\mathbb{F}$ into $\Omega_r$ [];
14:   return V, L, $\Omega_r$ []
```

C. VSNFS EMBEDDING PROCEDURE

The basic PSO algorithm has been proved that it can efficiently address the optimization problem in the continuous domain [43]. However, the VSNFs embedding problem belongs to the category of the discrete optimization problem. The VSNFs embedding model needs to be redefined when applying the basic PSO algorithm into the VSNF embedding problem. The modifications of the basic PSO based VSNF embedding algorithm can be described as follows.

[i] Position of Particle. Each VSNF along the SSC needs to be embedded onto a unique substrate node. Let the λ to represent the dimension of search space, i.e., λ represents the number of VSNFs along the SSC requests. Correspondingly, $X_i(t) = [x_{i,1}, x_{i,2}, \dots, x_{i,j}, \dots, x_{i,\lambda}]$ is the position of the ith VSNF embedding solution at the tth iteration, where $x_{i,j}$ is positive integer that represents the index value of the jth VSNF embedding onto the substrate network.

[ii] Speed of Particle. In order to guide the particle for better solution, let $V_i(t) = [v_{i,1}, v_{i,2}, \dots, v_{i,j}, \dots, v_{i,\lambda}]$ represent the speed of the ith VSNF embedding solution at the tth iteration, where $v_{i,j}$ is a binary variable. It is 1 (true) if the jth VSNF along the SSC request reselects the substrate node from $n_i^s \in \mathbb{N}^s$; and 0 (false) otherwise.

[iii] Fitness Function. According to the formulated ILP model, we set the objective function as the fitness function, which can be denoted by $f[X]$.

[iv] Dynamic Updating of Particle. According to the definitions regarding the position and the speed of particle, the updating function can be expressed as

$$\begin{cases} V_i(t+1) = \omega V_i(t) \oplus cr_1(X_{pbest_i}(t) \Theta X_i(t)) \oplus cr_2(X_{Gbest}(t) \Theta X_i(t)) \\ X_i(t+1) = X_i(t) \oplus V_i(t+1) \end{cases} \tag{15}$$

where $X_{pbest_i}(t)$, $X_{Gbest}(t)$ are both λ-dimensional vectors, which represent the local optimal solution of the ith VSNF embedding at the tth iteration, and the global optimal solution of the ith VSNF embedding at the tth iteration, respectively. ω is the inertia weighting factor that denotes the probability that the particle maintains its current speed, and parameters cr_1 and cr_2 both represent the acceleration factors, which are used to adjust the probability based on the $X_{pbest_i}(t)$ and $X_{Gbest}(t)$. Note that ω, cr_1 and cr_2 are all positive constants, and satisfy the relationship $\omega+cr_1+cr_2=1$. In the iterative process, the ith VSNF would be dynamically updated to adjust the $V_i(t+1)$ based on the $X_{pbest_i}(t)$, $X_{Gbest}(t)$ and $V_i(t)$, thus updating the "best so far" VSNF embedding solution $X_i(t)$.

[v] Definitions of Operations. The addition operation $\oplus$ and the subtraction operation Θ need to be redefined based on the SSC-DMP. We are running an intuitive example to illustrate the definitions of Θ and $\oplus$ based on the Eq. (15). For example, $X_{pbest_i}(t) \Theta X_i(t)$ denotes the difference between the current VSNF embedding solution $X_i(t)$ and the local optimal VSNF embedding solution $X_{pbest_i}(t)$. If $X_i(t)$ and $X_{pbest_i}(t)$ have the same value in the same

dimension, the $X_{pbest_i}(t)\Theta X_i(t)$ equals 0; and 1 otherwise. In addition, $X_i(t)\oplus V_i(t+1)$ is used to obtain the adjustment solution for the VSNF embedding.

The design of the proposed VSNF embedding algorithm can be depicted in **Algorithm 3**. The proposed VSNF embedding algorithm based on the basic PSO takes the $f\,[X_r]$=*Obj* as the optimization objective, where $X_r=\{x_1^k, x_2^l, ..., x_i^m\}$ is the possible VSNF embedding solution for the $r \in \mathbb{R}$, where x_i^m represents the substrate node $n_i^s \in \mathbb{N}^s$ being the location of VSNF $v_m \in \mathbb{F}$. Specifically, we firstly initialize the position vectors X_{pbest_i} and X_{Gbest} of **Algorithm 3** (***Lines* 3-4**). After that, the VSNF location selection function is adopted to update the position and speed of the particle (***Line* 8**). Next, **Algorithm 3** checks whether to satisfy the site capacity constraints, and then the delay, connection constraints are also checked (***Line* 9**). If so, update the position $x_{i,j}$ and the speed $v_{i,j}$ of the particle; otherwise, regenerate the position $x_{i,j}$ and the speed $v_{i,j}$ of the particle, and recalculate the fitness function value $f\,[X_i]$ (***Lines* 10-12**). As for each particle of the particle swarm, if $f\,[X_i] < f\,[X_{pbest_i}]$, then X_{pbest_i}=X_i. In addition, if $f\,[X_{pbest_i}] < f\,[X_{Gbest}]$, then $X_{Gbest}=X_{pbest_i}$ (***Lines* 13-16**). We finally check the current iterative number and determine whether $f\,[X_{Gbest}]>0$. If so, $f\,[X_r]$ is the total security resource consumption, and **Algorithm 3** can output the VSNF embedding solution $X_r=\{x_1^k, x_2^l, ..., x_i^m\}$; otherwise, $f\,[X_r]\leftarrow +\infty$ would be set (***Lines* 18-19**).

Algorithm 3 VSNF embedding algorithm for determining the location of each VSNF

```
1: Input: Substrate network graph $\mathbb{G}=(\mathbb{N}^s,\mathbb{E}^s,A_n,A_e)$, SSC request
   $\mathbb{G}^v=(\mathbb{N}^v,\mathbb{E}^v)$ and available VSNFI set $\Omega_r$;
2: Output: $X_r=\{x_1^k, x_2^l, ..., x_i^m\}$, $f\,[X_r]$;
3: Initialize the position vector of the ith VSNF $X_{pbest_i}$;
4: Initialize the position vector with the best fitness $X_{Gbest}$;
5: while ($t_{max}$=MG) do
6:   foreach request $r_k \in \mathbb{R}$ do
7:     foreach VSNF $v_l \in \mathbb{F}$ along the $r_k \in \mathbb{R}$ do
8:       function Selection_VSNFlocation $(v_l, v_n, C(v_l^{se_l}))$
9:       if IsResourceAvailable $(i, l, \mathbb{R})$ then
10:          Update the position $x_{i,j}$ and the speed $v_{i,j}$ of the
             particle according to (15);
11:      else if
12:          Regenerate the $x_{i,j}$ and the $v_{i,j}$ of the particle, and
             recalculate the $f\,[X_i]$;
13:          if $f\,[X_i] < f\,[X_{best_i}]$ then
14:              $X_{pbest_i} \leftarrow X_i$;
15:          if $f\,[X_{pbest_i}] < f\,[X_{Gbest}]$ then
16:              $X_{Gbest} \leftarrow X_{pbest_i}$;
17:      update ResourceAvailable $(i, l, \mathbb{R})$
18: if $f\,[X_{Gbest}]>0$ then
19:   return $f\,[X_r]$, $X_r=\{x_1^k, x_2^l, ..., x_i^m\}$
```

D. OVERALL PROCEDURE

To sum up, the whole procedure of the proposed PSO algorithm is presented in Fig. 6. Our proposed PSO algorithm can be divided into three main phases, including the security path selection phase, security level determination phase and VSNF embedding phase. Here, we employ the big $\mathcal{O}$ notation to show the time complexity of the entire proposed PSO algorithm. Let $|\mathbb{F}|$ and $|\mathbb{N}^s|$ respectively denote the number of the VSNFs and the substrate nodes in the cloud datacenter networks. The computational complexity of the proposed PSO algorithm can be analyzed like this. The time complexity of PSO is evaluated based on the following remarks: *(i)* The PSO algorithm performs the *k*-Dijkstra algorithm to find the shortest security paths, and the complexity of the phase is $\mathcal{O}(k+|\mathbb{F}|\log|\mathbb{F}|+|\mathbb{N}^s|)$; *(ii)* To obtain the VSNFI set *Ω_r []* that satisfies the security level requirements, the VSNFI set *Ω_r []* can be obtained using the SLSA algorithm, the time complexity of which is $\mathcal{O}(|\mathbb{F}|)$; *(iii)* The VSNF embedding algorithm is adopted to obtain the location of the VSNFs, which runs the $\mathcal{O}(MG|\mathbb{F}|)$ time. Based on these remarks mentioned above, the entire complexity of PSO is $\mathcal{O}(k+|\mathbb{N}^s|+|\mathbb{F}|\log|\mathbb{F}|+MG|\mathbb{F}|+|\mathbb{F}|)$, where *MG* is the maximum iteration number of the PSO algorithm. As $|\mathbb{F}|<<|\mathbb{N}^s|$, the complexity of the PSO algorithm is expressed as $\mathcal{O}(|\mathbb{N}^s|+|\mathbb{F}|\log|\mathbb{F}|+MG|\mathbb{F}|)$ .Thus, the proposed PSO algorithm belongs to the polynomial time algorithm that is appropriate for dynamic and online SSC-DMP under the cloud datacenter networks.

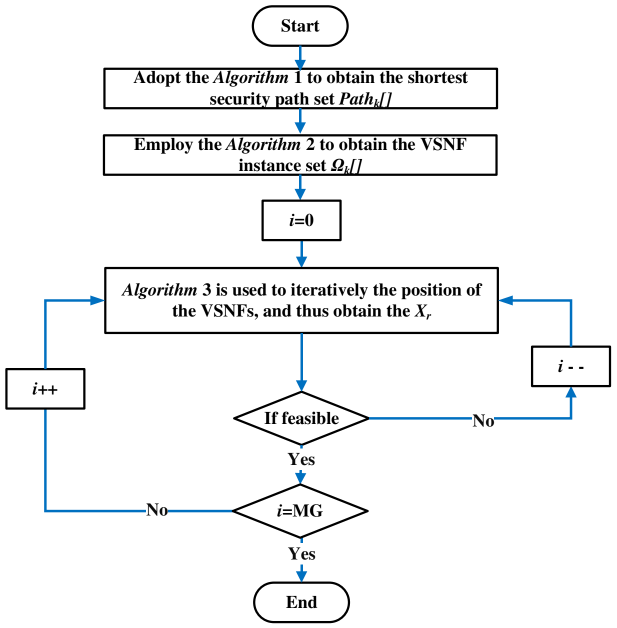

Start
Adopt the Algorithm 1 to obtain the shortest security path set Path_k[]
Employ the Algorithm 2 to obtain the VSNF instance set Ω_k[]
i=0
Algorithm 3 is used to iteratively the position of the VSNFs, and thus obtain the X_r
i++
i - -
If feasible
No
Yes
i=MG
No
Yes
End

FIGURE 6. Process flow of the PSO algorithm

For the dynamic deployment of the PSO approach, we consider the SSC resource allocation node in the NFV-MANO. Note that the proposed PSO approach runs in the NFV-MANO node, which interacts with the substrate cloud servers via the network telemetry technologies (such as P4, Net vision, etc.) [4], and help allocate security service resource in the cloud datacenter networks. Once the PSO algorithm gradually reaches the convergence state, this algorithm can rapidly make right control strategy, according to the real-time network state and dynamic online SSC requests. The NFV-MANO node can receive SSC requests from mobile users and construct different SSCs with different VSNFIs. At each time step, the proposed PSO approach in the NFV-MANO can dynamically observe the network state of the cloud datacenter networks, and percept online SSC requests from mobile users. Then, PSO can make a decision. Through the online evolution process, the NFV-MANO would determine an efficient SSC embedding strategy, i.e., the locations of the VSNFs $X_r = \{x_1^k, x_2^l, ..., x_i^m\}$ and their corresponding security service paths $\mathbb{P}$.

VI. EVALUATION

In this section, the cloud datacenter network simulation environment is fully described, and then we analyze the simulation results.

A. SIMULATION SETUP

As for the substrate networks, similar to our prior work [3], we consider the cloud datacenter network simulation context. In this subsection, we conduct the simulations to evaluate the performance of the proposed PSO algorithm on top of the real-world topologies, i.e., Internet2 research network including 12 nodes and 15 links, and Cogentco including 197 nodes and 245 links. Note that we choose such a small-scale Internet2 network topology, this is because we can obtain the ILP's solution within a polynomial time, and thus evaluate how close the proposed PSO algorithm are close to the ILP's solution. However, with the increase of network scale, the computation cost using the exact tools to address the ILP model is unaffordable. In order to further evaluate the effectivity of the proposed PSO algorithm, a large-scale Cogentco network topology is selected as the network topology. As for the substrate parameter configuration, the choice of these simulation parameters as well as their distribution are motivated by simulations and evaluations of a well-studied and related literatures [44, 45]. In both networks, we configure the substrate nodes in the network with [200, 800] *units* of available IT resource capacities. The number of VSNFs contained in each edge server is within [1, 8], each of which has 10 security function types. Each VSNF instance in the substrate network has its corresponding security level, which is randomly distributed in [1, 6]. As for each link between the edge servers, the bandwidth resource capacity is uniformly within [0, 2000] *Mbps*. As for the security service requests, we consider each request consists of a set of VSNFs varying from 2 to 5, each of which is picked from 10 security function types. The amount of requested IT resources of a type VSNFI $v_x \in \mathbb{F}$ per security level is uniformly within [20, 30] *units*. In addition, the flow rate of security service chain is uniformly within [10, 60] *Mbps*, and the deadline are randomly selected within [20, 50] *ms*. The security level of the requested VSNF is randomly distributed in [1, 6]. To efficiently make the PSO algorithm converge in a fast optimal manner, according to the literature [43], the number of the particles *Num* in particle swarm is fixed to 50, the maximum iterative number *MG* of the PSO algorithm is fixed to 1000. In (15), the controlling parameters ω, cr_1 and cr_2 are empirically set for 0.1, 0.2 and 0.7, respectively. Besides, the weighting factors ω_1 and ω_2 are both fixed to 1. The simulations are operated on a PC with a 3.6 GHz two core Intel® Core^{TM} i7 and 8GB RAM. In the next subsection, we present and discuss the simulation results.

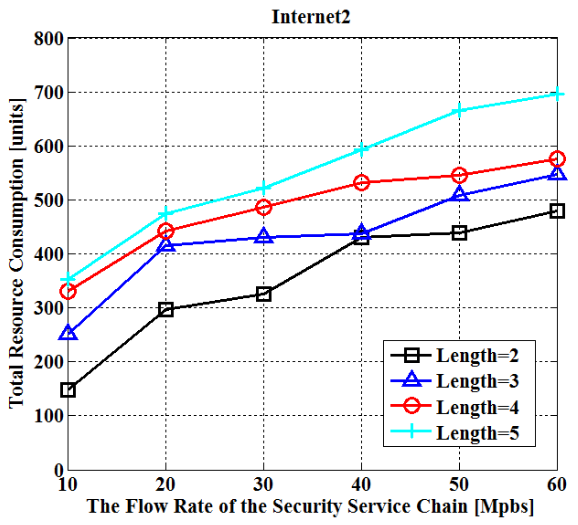
B. RESULTS AND DISCUSSIONS

In order to evaluate how close our provided solutions to the optimal one, the formulated ILP model is implemented in the Gurobi 6.0 optimizer [46]. To further demonstrate the effectivity of the proposed PSO algorithm, we compare our proposed algorithm with three other algorithms, and the compared approaches are all implemented in MATLAB. Since our proposed PSO algorithm consists of three parts, i.e., *k*-Dijkstra algorithm, the security level selection algorithm and the VSNF embedding algorithm, we combine some previous classical algorithms (e.g., greedy biased approaches [22], tabu search (TS)-based approach [32]) to generate the other three algorithms. The details are as follows.

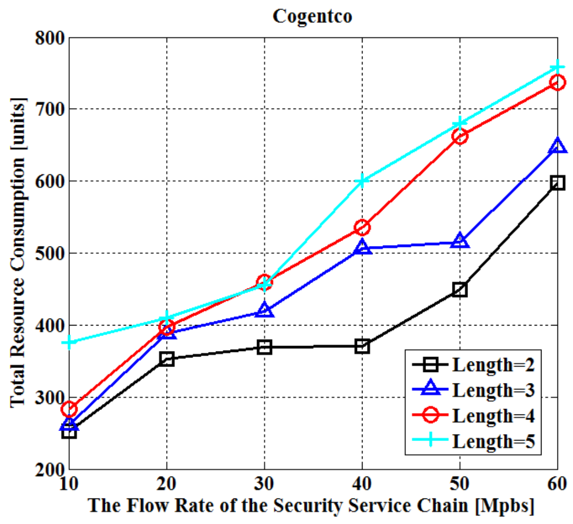
1) **Tabu Search based SSC embedding (denoted as TS):** in the first stage, it finds the shortest path by shortest path heuristic, and in the second stage, it embeds the VSNFs along the shortest path by the tabu search algorithm.
2) **Greedy embedding with bias towards Least Loaded (denoted as GLL):** it selects the feasible path with the shortest path heuristic and employs the Greedy embedding with bias least loaded to embed the VSNFs along the shortest path.
3) **Greedy embedding with bias towards Fast Processing (denoted as GFP):** it employs the shortest path heuristic to determine the shortest path and uses the Greedy embedding with bias fast processing to assign the VSNFs.

In order to ensure the sufficient statistical accuracy, we obtain each data point by averaging the obtained results from 20 independent simulations. The obtained simulation results and the corresponding analysis are as follows.

Metric1: The Total Resource Consumption. This is the objective of our formulated model, which can be calculated by the sum of the security IT resource consumption on the substrate node and the bandwidth on the substrate link. Note that the IT resource consumption is mainly determined by the length of the SSC request, and the bandwidth consumption is determined by the flow rate of the SSC request. Fig. 7 shows the total resource consumption of the proposed PSO algorithm by varying the increasing flow rate of the SSC request. As shown in Fig. 7, the curve of each length becomes rising, with the increasing flow rate of the SSC request. The reason is that the flow rate of the SSC request mainly affects the bandwidth consumption, and larger amount of the flow rates would cause more resource consumption. From another point of view, we observe that the length of 2 has the lowest resource consumption, while the length of 5 has the highest resource consumption. This is because the longer length of the SSC request would result in more IT resource consumption, which could further increase the total resource consumption. This phenomenon further verifies the fact that the total resource consumption could simultaneously be affected by the length of the SSC request, and the flow rate of the SSC request.



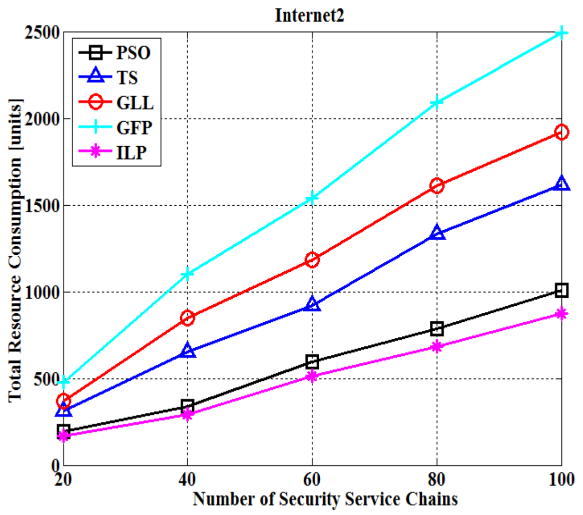
(a) Internet2 with 12 nodes



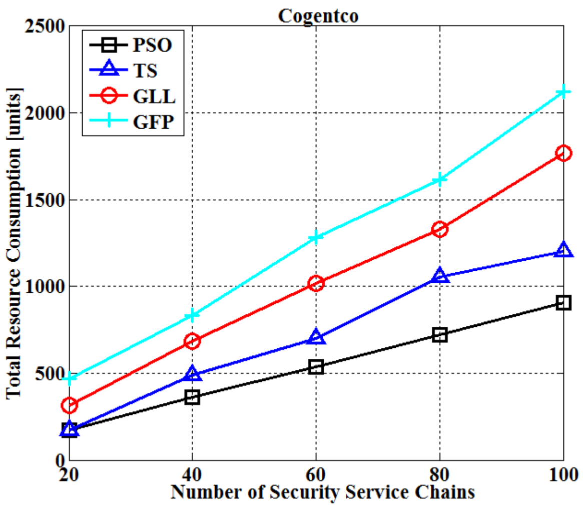
(b) Cogentco with 197 nodes

FIGURE 7. Impact relationship between the length and the flow rate of security service chain

Next, we compare our proposed PSO algorithm with the baseline algorithms (i.e., the GFP, GLL and TS algorithm) by varying the number of the SSC requests. Fig. 8 shows the variations of the total resource consumption with service arrivals. As shown in Fig. 8 (a), the mean resource consumption of the ILP's solution is 497.5 *units*, the mean resource consumption of the proposed PSO algorithm is 526.4 *units*, the mean resource consumption of the TS algorithm is 967.3 *units*, the mean resource consumption of the GLL algorithm is 1316.5 *units* and the mean resource consumption of the GFP algorithm is 1536.7 *units*. It can be observed that the ILP's solution can obtain the minimum objective value because the Gurobi 6.0 optimizer of solving the ILP model belongs to the category of exact solution, which can exactly calculate the optimal objective value at the expense of execution time. Besides, the PSO algorithm, the TS algorithm, the GLL algorithm and the GFP algorithm provide the upper-bound on the optimization of the total resource consumption. The reason why the proposed PSO algorithm is the closest to the ILP's solution can be because PSO makes it possible to obtain more efficient solution in the dynamics, thanks to the advantage of the evolutionary computation. Moreover, the superiority of PSO becomes more apparent when applying to the large-scale network scenarios, as depicted in Fig. 8 (b). Compared to the TS algorithm, the GLL algorithm and the GFP algorithm, PSO can averagely reduce the resource consumption of approximately 313.8 *units*, 686.3 *units* and 895.2 *units* on the Internet2 research network, respectively, and approximately 246.8 *units*, 489.2 *units* and 829.5 *units* on the Cogentco network, respectively. The reasons behind this are apparent: First of all, k feasible security service paths are selected by employing the k-Dijkstra algorithm, which can make a global decision based on the feasible security paths. Secondly, the SLSA algorithm is used to select the VSNFs with appropriate security level. Finally, the VSNFs are embedded along the best paths using the PSO algorithm to minimize the resource consumption. As noted from Fig. 8 (a) and Fig. 8 (b), the reason why the TS algorithm only performs worse than the PSO algorithm could be because the TS algorithm is the application of the original TS algorithm, however, the original TS algorithm is strongly dependent on the initial solution and controlling parameters, which easily reduces the convergence speed and fails to make a global decision. Two algorithms based on the greedy biased approaches, i.e., GFP, GLL, perform the worst among the compared algorithms, the reason is that the greedy heuristics have a rapid convergence at the price of the risk of sticking at a local minimum that would not be effective. Eventually, it is observed that GLL performs better than GFP. This is because GFP are likely to have shorter queues, which indicates that the SSC requests embedded onto such nodes get processed earlier, and hence they do not occupy the node resources for longer periods which could reduce the resource consumption. In conclusion, the proposed PSO algorithm can satisfy the requirements

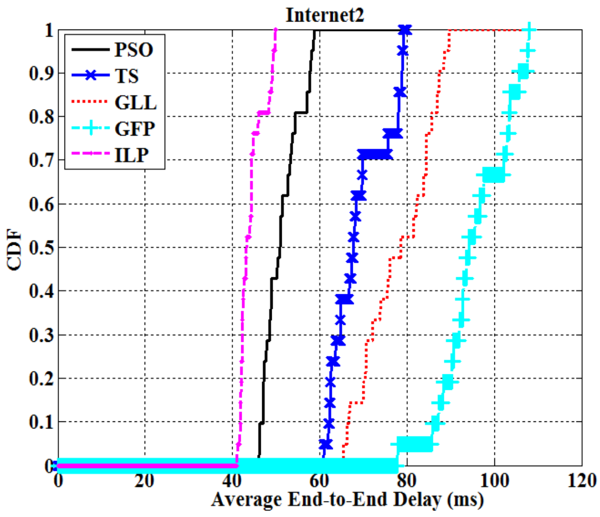


(a) Internet2 with 12 nodes

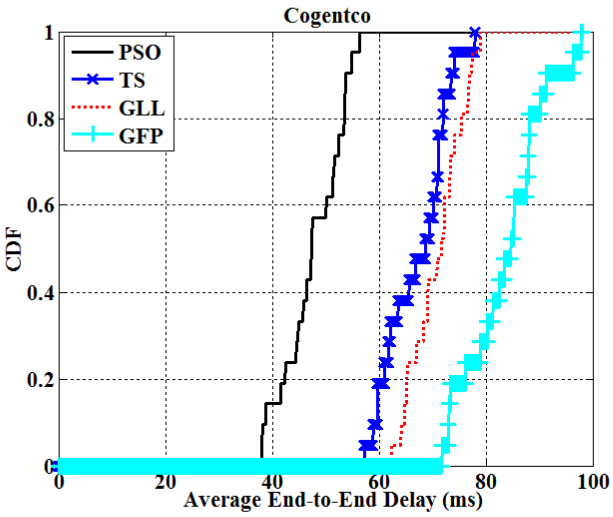


(b) Cogentco with 197 nodes

FIGURE 8. Results on total resource consumption (varying number of security service chain from 20 to 100)



(a) Internet2 with 12 nodes



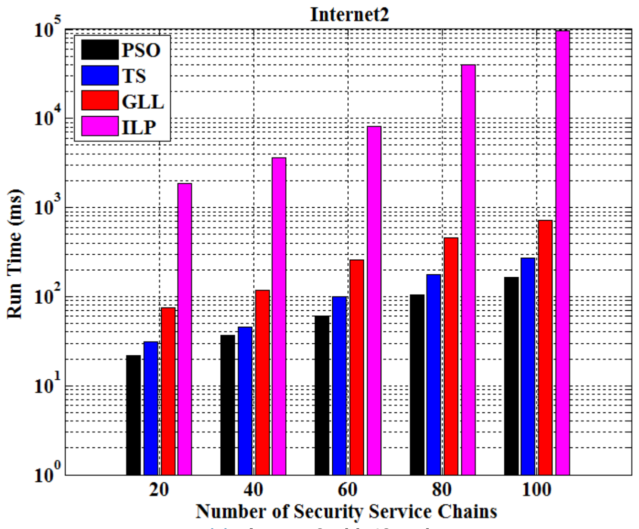
(b) Cogentco with 197 nodes

FIGURE 9. Results on CDF of average delay (varying number of security service chain from 20 to 100)

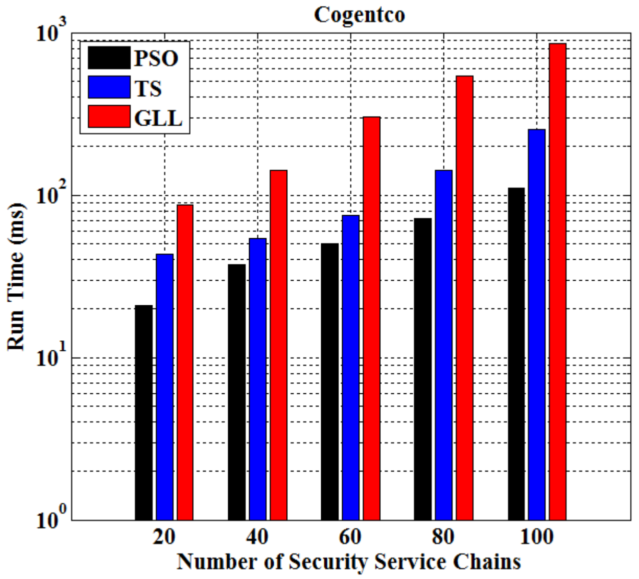
of the security level, precisely achieve the dynamic allocation of the SSC requests, and consume more security substrate resources. This further demonstrates that our proposal can effectively save more substrate resources for the cloud datacenter networks.

Metric2: Average End-to-End Latency. This metric is the sum of the processing time of all VSNFs on the critical path and the transmission time of all links on the service function path. Fig. 9 shows the CDF of the end-to-end latency by varying the increasing number of SSC requests. Generally, we can observe in Fig. 9 that the end-to-end latency of the security function path is approximately within [45, 105] *ms*. More specifically, as depicted in Fig. 9 (a), we can notice an expected phenomenon that the ILP's solution outperforms the compared algorithms and incurs the minimum end-to-end latency. Then, compared with the TS algorithm, the GLL algorithm and the GFP algorithm, the proposed PSO algorithm can achieve the minimum end-to-end latency, and reduce the average end-to-end delay down to approximately 20.6%, 31.2% and 38.5%, respectively. The reason is that our proposal is based on the basic PSO algorithm to determine all feasible security service paths, so as to further make the global SSC resource allocation strategy. Note that three compared benchmark algorithms (i.e., TS, GLL and GFP) fail to follow the trend of the proposed PSO algorithm, this is because the proposed PSO algorithm simultaneously considers the transmission delay and the available bandwidth on the substrate links when selecting the security function paths. Next, the fact that TS performs better than GLL and GFP could be because TS has a chance to iteratively improve the solution. Finally, the reason why the GFP algorithm performs worst among the compared algorithms in terms of the end-to-end latency could be because GFP always embeds a given VSNF to the node that processes it faster. This indicates that a node that has least processing time for a given VSNF is likely to always be overloaded, resulting in a longer queue at such a node, which implies that the other VSNFs for the same SSC request should wait for such a function. Fig. 9 (b) demonstrates the CDFs of the recorded end-to-end delay over all the iterations traced, and show the similar performance of the end-to-end delay as Fig. 9 (a) has shown. The explanation is also same and thus omitted here.

Metric3: Algorithm Run Time. The algorithm run time represents the decision time required before each SSC request is successfully embedded onto the substrate networks. Moreover, the algorithm run time can be regarded as a crucial metric in the SSC resource allocation. According to the previous computational complexity analysis mentioned in **Section V**, the algorithm run time is determined by the length of SSC $|\mathbb{F}|$ and the number of the substrate node $|\mathbb{N}^s|$. Fig. 10 (a) and Fig. 10 (b) show the algorithm run time for different number of SSC under the Internet2 and Cogentco network topology. Note that we omit the curves of GFP, since it has the similar algorithm run time with GLL. Towards the end of the simulation (at approximately 100 SSC requests) in Fig. 10 (a), the algorithm run time of the ILP's solution is 92115.6 *ms*, the run time of the proposed PSO algorithm is 174.1 *ms*, the run time of the TS algorithm is 273.6 *ms*, and the run time of the GFP algorithm is 686.9 *ms*. The run time of the TS algorithm is mainly determined by the length of SSC $|\mathbb{F}|$ and the number of the substrate node $|\mathbb{N}^s|$, which performs the $\mathcal{O}_{\text{TS}}=\mathcal{O}\left(|\mathbb{F}|^2+|\mathbb{F}|\lambda+|\mathbb{N}^s|^2\right)$ computations (for details, refer to [32]), where λ is the length of the Tabu list. Then, based on the ILP model for SSC-DMP, both Boolean variables and nonlinear constraints in the model are interactable for the large-scale network scenarios, with the computational complexity of $\mathcal{O}_{\text{ILP}}=\mathcal{O}(2^{|\mathbb{N}^s|^4})$, and the formulated ILP based SSC-DMP can not be addressed in the acceptable time scales for the realistic network problem dimensions. Thus, the run time of the ILP's solution is the highest among the compared algorithms and we find it hard to address the SSC-DMP when applying the ILP to the Cogentco network. Eventually, the reason why the proposed PSO algorithm performs better than the GFP algorithm is because the overall computational complexity of the GFP algorithm is $\mathcal{O}_{\text{GLL}}=\mathcal{O}(|\mathbb{N}^s|^2+|\mathbb{N}^s|\log|\mathbb{N}^s|)$ (for details, refer to [6]), however, our proposal employs the *k*-Dijkstra algorithm, the SLSA algorithm and the VSNF embedding algorithm, which performs the computations of $\mathcal{O}_{\text{PSO}}=\mathcal{O}(|\mathbb{N}^s|+|\mathbb{F}|\log|\mathbb{F}|+MG|\mathbb{F}|)$. Fig. 10 (b) demonstrates the algorithm run time with the increasing number of the SSC requests, and show the similar performance of the algorithm run time as Fig. 10 (a) has shown. The explanation is also same and thus omitted here. Combined with the theoretical analysis, the trend of each algorithm is basically consistent with the theoretical analysis. The execution time can be reduced if we use more powerful computation platform other than a personal computer, as in the common case of a practical NC&M system. Hence, the proposed PSO algorithm in this paper can efficiently obtain a feasible solution even in quite a large-scale network considering the characteristics of the realistic cloud datacenter network topologies and SSC requests.



(a) Internet2 with 12 nodes



(b) Cogentco with 197 nodes

FIGURE 10. Results on algorithm run time (varying number of security service chain from 20 to 100)

C. SECURITY ANALYSIS

During the process of SSC embedding oriented to the cloud datacenter networks, the proposed PSO algorithm can determine the placement of the VSNFs and the routing of the traffic flows, according to the SSC requests and the specific security levels of the requested VSNFs. The TS, GLL and GFP algorithm fail to consider the specific security level of VSNFs, and lack of the perception for the network security situations. For the SSC requests, the TS, GLL and GFP algorithm determine the VSNFs with the highest security levels, and thus consume more security resources fragments and result in more SSC embedding overhead. However, our proposed PSO algorithm can dynamically determine the security levels of VSNFs. More specifically, when facing higher network security risks, PSO can instantiate the VSNFs with the higher security levels. On the contrary, the VSNF with the lower security level could correspondingly be determined. In conclusion, the proposed PSO algorithm can dynamically and flexibly schedule or release security resources, based on the network security situation and the substrate network state. Moreover, PSO can satisfy the security service requirements while significantly reducing the security resource consumption.

VII. CONCLUSION

In this paper, we investigate the problem of VSNF chain dynamic embedding for the cloud datacenter networks. The proposed approach, called PSO, fully considers the specific security level and delay requirements of security applications, while guaranteeing that the IT resources and bandwidth for cloud datacenter networks can be accurately utilized. We firstly present the network architecture of the SSC dynamic embedding. More specifically, we introduce a security level protection-based SSC embedding network architecture and present the three-level based security service model. Then, we formulate the SSC-DMP as an ILP model and devise a meta-heuristic-based SSC dynamic embedding algorithm. In the first stage, we adopt the *k*-Dijkstra algorithm to all the pre-routing security function paths. Based on the selected security function paths, in the second stage, we design the SLSA algorithm to determine the VSNFI set that can satisfy the specific security service level of the requested VSNFs. In the last stage, we design a VSNF embedding algorithm based on the particle swarm optimization to assign the VSNFs with minimum security resource consumption. Finally, we conduct simulations on two scale realistic network topologies. The obtained simulation results demonstrate the proposed PSO algorithm outperforms three other mainstreaming algorithms in terms of the total resource consumption, the end-to-end delay and algorithm run time.

This work can be extended in several directions. Firstly, the proposed PSO algorithm mainly depends on the online information of substrate resource utilization and security service demands, which still remains no guarantee that the results obtained by the proposed PSO algorithm are the most accurate in the dynamics. We are still exploring how to combine our proposed PSO algorithm with some machine-learning-based approaches (e.g., GNN [7], MLP [47, 48], LSTM [49]), which can forecast the future trends of link/node load and traffic variations ahead of time, and thus improve the efficacy of the SSC resource allocation algorithm. Second, we have conducted evaluations on the simulators. In the future, we will try to address how to fill the gap between simulator and the real edge-cloud network platforms for the proposed PSO algorithm implementation.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 62071483. The authors would like to thank the editor and anonymous expert reviewers for the instructive comments, and we appreciate your warm work earnestly.

REFERENCES

[1] D. Bhamare, R. Jain, M. Samaka, et al., “A survey on service function chaining,” *J Netw Comput Appl*. vol.75, pp. 138-155, 2016.

[2] Y. Liu, Y. Lu, X. Li, W. Qiao, Z. Li and D. Zhao., “SFC Embedding Meets Machine Learning: Deep Reinforcement Learning Approaches,” *IEEE Communications Letters*, doi: 10.1109/LCOMM.2021.3061991.

[3] W. Qiao, Y. Liu, Y. Lu, et al., “A Novel Approach for Service Function Chain Embedding in Cloud Datacenter Networks,” *IEEE Communications Letters*, 2020. doi: 10.1109/LCOMM.2020.3043845.

[4] Y. Liu, H. Lu, X. Li, et al., “Dynamic Service Function Chain Orchestration for NFV/MEC-Enabled IoT Networks: A Deep Reinforcement Learning Approach,” *IEEE Internet of Things Journal*, 2020. doi: 10.1109/JIOT.2020.3038793.

[5] Y. Liu, et al., “GDM: A General Distributed Method for Cross-Domain Service Function Chain Embedding,” *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1446-1459, Sept. 2020.

[6] Y. Liu, Y. Lu, W. Qiao and X. Chen, “A Dynamic Composition Mechanism of Security Service Chaining Oriented to SDN/NFV-Enabled Networks,” *IEEE Access*, vol. 6, pp. 53918-53929, 2018.

[7] Y. Liu, et al., “On Dynamic Service Function Chain Reconfiguration in IoT Networks,” *IEEE Internet of Things Journal*, vol. 7, no. 11, pp. 10969-10984, Nov. 2020.

[8] R. Mijumbi et al., “Network Function Virtualization: State-of-the-Art and Research Challenges,” *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[9] X. Fei, et al., “FlexNFV: Flexible Network Service Chaining with Dynamic Scaling,” *IEEE Network*, vol. 34, no. 4, pp. 203-209, 2020.

[10] Q. Zhang, et al., “Adaptive Interference-Aware VNF Placement for Service-Customized 5G Network Slices,” *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, Paris, France*, 2019, pp. 2449-2457.

[11] V. Eramo, et al., “An Approach for Service Function Chain Routing and Virtual Function Network Instance Migration in Network Function Virtualization Architectures,” *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 2008-2025, 2017.

[12] ETSI, “Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV,” https://www.etsi.org/deliver/etsigs/NFV/001099/003/01.02.0160/gsNFV003v010201p.pdf.

[13] R. Doriguzzi-Corin, et al., “Application-Centric Provisioning of Virtual Security Network Functions,” *Proc. of the Third IEEE International Workshop on Security in NFV-SDN (SN-2017)*, 2017.

[14] R. Doriguzzi-Corin, et al., “Dynamic and Application-Aware Provisioning of Chained Virtual Security Network Functions,” *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 294-307, 2020.

[15] S. Yang, et al., “Recent Advances of Resource Allocation in Network Function Virtualization,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 2, pp. 295-314, 2021.

[16] J. Gil Herrera and J. F. Botero, “Resource Allocation in NFV: A Comprehensive Survey,” *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518-532, Sept. 2016.

[17] P. Jin, et al., “Latency-aware VNF Chain Deployment with Efficient Resource Reuse at Network Edge,” *IEEE INFOCOM 2020 - IEEE*

Conference on Computer Communications, Toronto, ON, Canada, 2020, pp. 267-276.

[18] Q. Zhang, et al., “Joint Optimization of Chain Placement and Request Scheduling for Network Function Virtualization,” *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS),* Atlanta, GA, 2017, pp. 731-741.

[19] X. Fei, et al., “Towards load-balanced VNF assignment in geo-distributed NFV Infrastructure,” *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, Vilanova i la Geltru, pp. 1-10, 2017.

[20] T. Wang, et al., “Multi-resource Load Balancing for Virtual Network Functions,” *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, Atlanta, GA, pp. 1322-1332, 2017.

[21] J. Luo, et al., “SDN/NFV-Based Security Service Function Tree for Cloud,” *IEEE Access*, vol. 8, pp. 38538-38545, 2020.

[22] D. Bhamare, M. Samaka, A. Erbad, *et al.*, “Exploring microservices for enhancing internet QoS,” *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, pp. e3445, 2018.

[23] M. Khoshkholghi, J. Taheri, D. Bhamare *et al.*, “Optimized Service Chain Placement Using Genetic Algorithm,” *2019 IEEE Conference on Network Softwarization (NetSoft)*, Paris, France, 2019, pp. 472-479, doi: 10.1109/NETSOFT.2019.8806644.

[24] H. Chen, et al. “MOSC: a method to assign the outsourcing of service function chain across multiple clouds,” *Computer Networks*, vol. 133, pp. 166-182, 2018.

[25] L. Qu, et al. “Reliability-Aware Service Chaining In Carrier-Grade Softwarized Networks,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 558-573, March 2018.

[26] J. Fan, et al. “Availability-aware mapping of service function chains,” *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, Atlanta, GA, 2017, pp. 1-9.

[27] G. Ishigaki, et al. “DeepPR: Progressive Recovery for Interdependent VNFs With Deep Reinforcement Learning,” *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2386-239.

[28] Y. Liu, Y. Lu, W. Qiao, et al., “Reliability-Aware Service Chaining Mapping in NFV-Enabled Networks,” *ETRI Journal*, vol. 42, no. 2, pp. 207-233, 2019.

[29] H. R. Khezri, et al., “Deep Reinforcement Learning for Dynamic Reliability Aware NFV-Based Service Provisioning,” *2019 IEEE Global Communications Conference (GLOBECOM)*, Waikoloa, HI, USA, 2019, pp. 1-6.

[30] F. Carpio, et al., “Improving reliability of service function chains with combined VNF migrations and replications,” arXiv:1711.08965, 2017.

[31] S. W. Shin, P. Porras and V. Yegneswara, “Fresco: Modular composable security services for software-defined networks,” *Proc. Netw. Distrib. Secur. Symp.*, 2013.

[32] Y. Liu, et al., “A New Approach for Delivering Customized Security Everywhere: Security Service Chain,” *Secur. Commun. Netw.*, vol. 2017, no. 3, 2017.

[33] A. Shameli-Sendi, Y. Jarraya, M. Pourzandi and M. Cheriet, “Efficient Provisioning of Security Service Function Chaining Using Network Security Defense Patterns,” *IEEE Transactions on Services Computing*, vol. 12, no. 4, pp. 534-549, 1 July-Aug. 2019.

[34] *Information Processing Systems—Open Systems Interconnection OSI Part 1–5 DIS 9646*, 1989.

[35] X. Li, M. Samaka, H. A. Chan, D. Bhamare, L. Gupta, C. Guo, et al, “Network slicing for 5G: Challenges and opportunities,” *IEEE Internet Comput.*, vol. 21, no. 5, pp. 20-27, Sep. 2017.

[36] N. Jalodia, S. Henna and A. Davy, “Deep Reinforcement Learning for Topology-Aware VNF Resource Prediction in NFV Environments,” *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Dallas, TX, USA, 2019, pp. 1-5.

[37] Q. U. Cheng, “Based on level protection enterprise network security construction practice,” *Comput. Secur.*, 2011.

[38] C. Sun, et al., “NFP: Enabling network function parallelism in NFV,” in *ACM SIGCOMM*, 2017, pp. 43-56.

[39] Z. Wang, et al., “Service Function Chain Composition, Placement and Assignment in Data Centers,” *IEEE Trans. Netw. Service Manag,* vol. 16, no. 4, pp. 1638-1650, 2019.

[40] M. M. Tajiki, et al., “Joint Energy Efficient and QoS-Aware Path Allocation and VNF Placement for Service Function Chaining,” *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 374-388, March 2019.

[41] Y. Woldeyohannes, et al., “ClusPR: Balancing Multiple Objectives at Scale for NFV Resource Allocation,” *IEEE Trans. Netw. Service Manag*, vol. 15, no. 4, pp. 1307-1321, 2018.

[42] D. Bhamare, M. Samaka, A. Erbad, “Optimal virtual network function placement in multi-cloud service function chaining architecture,” *Computer Communications*, vol. 102, no. 3, pp. 1-16, 2017.

[43] N. Jin and Y. Rahmat-Samii, “Parallel particle swarm optimization and finite-difference time-domain (PSO/FDTD) algorithm for multiband and wide-band patch antenna designs,” *IEEE Transactions on Antennas and Propagation*, vol. 53, no. 11, pp. 3459-3468, Nov. 2005.

[44] F. Bari, et al., “Orchestrating virtual network functions”, *IEEE Trans. Netw. Service Mana,* vol. 13, no. 4, pp. 725-739, 2016.

[45] J. Wang, B. He, J. Wang and T. Li, “Intelligent VNFs Selection Based on Traffic Identification in Vehicular Cloud Networks,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4140-4147, May 2019.

[46] G. Optimization, http://www.gurobi.com/.

[47] X. Fei, et al., “Adaptive VNF Scaling and Flow Routing with Proactive Demand Prediction,” *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, Honolulu, HI, 2018, pp. 486-494.

[48] T. Subramanya, et al., “Machine Learning-Driven Service Function Chain Placement and Scaling in MEC-enabled 5G Networks”, *Computer Networks*, 2019. DOI: 10.1016/j.comnet.2019.106980.

[49] B. Li, et al., “Deep-learning-assisted network orchestration for on-demand and cost-effective vNF service chaining in inter-DC elastic optical networks,” *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, no. 10, pp. 29-41.

Wenxin Qiao received the B.S. degree in networking engineering from the Beijing University of Posts and Telecommunications (BUPT) in 2014, and the M.S. degree in software engineering from Shijiazhuang Campus, Army Engineering University, Shijiazhuang, China, where she is currently pursuing the Ph.D degree. Her research interests include key technologies of air-space-ground of integrated information communication network and service function chain resource allocation (SFC-RA).

Yicen Liu received the B.S. and M.S. degrees in communication and information system from Shijiazhuang Campus, Army Engineering University, Shijiazhuang, China, where he is currently pursuing the Ph.D. degree.

His current interests are key technologies of 6G communication, SDN/NFV/AI technology, and network slicing in edge clouds. He has published over 20 articles in refereed conferences and journals in the above areas. He has been invited to become a Reviewer of some top journals, such as *IEEE Journal on Selected Areas in Communications*, *IEEE Vehicular Technology Magazine, IEEE Communications Letters*, etc.

Leiping Xi received his Ph.D. degree in computer science and technology from Shijiazhuang Campus, Army Engineering University, China, in 2012. He is currently an Assistance Professor at Shijiazhuang Campus, Army Engineering University, China. His currently interests are the optimization of the UAV network resource management in edge computing and blockchain enabled UAV communication.

Xi Li was born in Shijiazhuang, China. He received his Ph.D. degree in computer science and technology from Shijiazhuang Campus, Army Engineering University, Shijiazhuang, China, in 2016, with his advisor Y. Lu. He is currently an Assistance Professor Shijiazhuang Campus, Army Engineering University, Shijiazhuang, China. His currently interest is the optimization of the network resource management in 5G networks.

Zhiwei Li was born in Shijiazhuang, China. He received the M.S. degree in communication and information system from the Shijiazhuang Campus, Army Engineering University, Shijiazhuang, China, in 2015, where he is currently pursuing the Ph.D. degree. His research interests are in the areas of communication schemes of UAVs network, network with deep reinforcement learning and software-defined UAV-enabled network.

Donghao Zhao received his MS degree in communication and information system from Shijiazhuang Campus, Army Engineering University, Shijiazhuang, China, in 2018, with his advisor Prof. Y. Lu. He is currently a lecturer in information and communication engineering department, Shijiazhuang Campus, Army Engineering University, Shijiazhuang, China. His current interests are in areas of SDN/NFV/AI, Service Function Chain Resource Allocation (SFC-RA) with AI and edge computing-enabled UAV networks.

Yu Lu received his Ph.D. degree in computer science and technology from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2001. He is currently a full professor Shijiazhuang Campus, Army Engineering University, Shijiazhuang, China. His current research primarily focuses on 5G/6G networks and software-defined security. He has published over 120 academic papers in several international journals and conference proceedings in the above areas.