Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Dynamic trust-based service function chain deployment method for disrupting attack chains*

Deqiang ZHOU[1], Xinsheng JI[†‡1,2], Wei YOU[1], Hang QIU[1], Jie YANG[1], Yu ZHAO[1], Mingyan XU[1]

[1]*Information Technology Institute, PLA Information Engineering University, Zhengzhou 450002, China*

[2]*Purple Mountain Laboratories, Nanjing 211111, China*

[†]E-mail: ndscjxs@126.com

Received Apr. 17, 2025; Revision accepted Oct. 15, 2025; Crosschecked Nov. 10, 2025; Published online Dec. 2, 2025

Abstract: Enhancement of service function chain (SFC) security ability by composing virtual network functions (VNFs) and allocating resources considering their security attributes can address the vulnerability threats in cloud environments, which is an important means of attempting to secure SFCs at the deployment stage. However, existing works do not consider the vulnerability correlation of the multi-step attack chains when completing SFC deployment based on trustworthiness. This results in existing security orchestration methods ignoring the differences in trustworthiness among network entities and focusing only on local trust optimization; these steps effectively disrupt the attack chains to secure SFCs. In this article, an innovative hierarchical trust model is proposed to assess the differentiated trustworthiness among network entities caused by vulnerability correlation. On the basis of trustworthiness assessment, both virtual trust of VNF combinations at the SFC composition stage and physical trust of physical node (PN) selections at the SFC placement stage are globally considered to disrupt the attack chains in SFCs as much as possible. To this end, the security-aware and cost-efficient SFC composition and placement (SCSCP) problem is formulated as an integer linear programming (ILP) problem, which is NP-hard. To tackle the SCSCP problem, the joint trust and cost global optimization (JTCGO) algorithm is proposed to dynamically update the trustworthiness and globally find the SFC deployment solutions including the VNF combination schemes and PN selection schemes. Simulation results demonstrate that our proposed algorithm can provide the optimal SFC deployment solutions for requests and can guarantee the SFC trustworthiness at a controllable cost, thereby protecting SFCs from network attacks in complex security environments.

Key words: Service function chain (SFC); Attack chain; Vulnerability correlation; Trustworthiness; SFC composition and placement

https://doi.org/10.1631/FITEE.2500218 **CLC number:** TP393

1 Introduction

The emergence of network function virtualization (NFV) (Herrera and Botero, 2016), along with software-defined networking (SDN), has led to profound changes in the networking paradigm. These changes have significantly transformed the way in which network services are deployed, managed, and operated. Thanks to NFV and SDN, any network service can be provided by a sequence of virtual network function (VNF) instances, which are interconnected via virtual links to form a service function chain (SFC). SFC simplifies the provision of on-demand services, facilitating applications in vertical industries, fifth-generation (5G) cellular network slicing, and the Internet of Things.

‡ Corresponding author

* Project supported by the National Key Research and Development Plan of China (No. 2022YFB2902204), the Key Research and Development Project of Henan Province (No. 231111211000), and the Top Talent Training Project of Henan Province (No. 244500510012)

ORCID: Deqiang ZHOU, https://orcid.org/0009-0002-0326-0513; Xinsheng JI, https://orcid.org/0009-0004-9579-6132

However, the reliance on NFV gives rise to more vulnerabilities due to infrastructure virtualization and resource sharing, thus expanding the attack surface of SFCs. A multitude of security risks are introduced into SFCs, including a set of security vulnerabilities inherited from NFV (Hasneen and Sadique, 2022) and those associated with the SFC-enabled domain (Pattaranantakul et al., 2023). Attackers lurking within the network can continuously explore and exploit vulnerabilities to launch attacks on the target VNFs. Their tactics range from surreptitiously implanting malicious software, to capitalizing on zero-day vulnerabilities for sudden raids, and to unleashing distributed denial of service (DDoS) attacks. As a result, SFCs are constantly under threat, and there is an urgent need for security methods to safeguard SFCs.

Nowadays, cyber attacks have generally evolved into multi-stage and multi-step cyber attack campaigns, that is, attack chains. In the NFV-enabled networks, attackers can launch network attacks on the target VNF in the virtual layer or the physical layer (Zhou et al., 2024). The attack chains in the virtual layer include VNFs connected to the target VNF, and the attack chains in the physical layer include the physical node (PN) where the target VNF is placed and the virtual machines on this PN. However, vulnerability correlation (Yu et al., 2013) indicates that to exploit a vulnerability in an attack chain, the result of exploiting the previous vulnerability in the attack chain must create the prerequisite conditions for exploiting this vulnerability. Therefore, the exploitation of vulnerabilities needs to satisfy specific prerequisite conditions to form an attack chain.

As shown in Fig. 1a, attackers can exploit

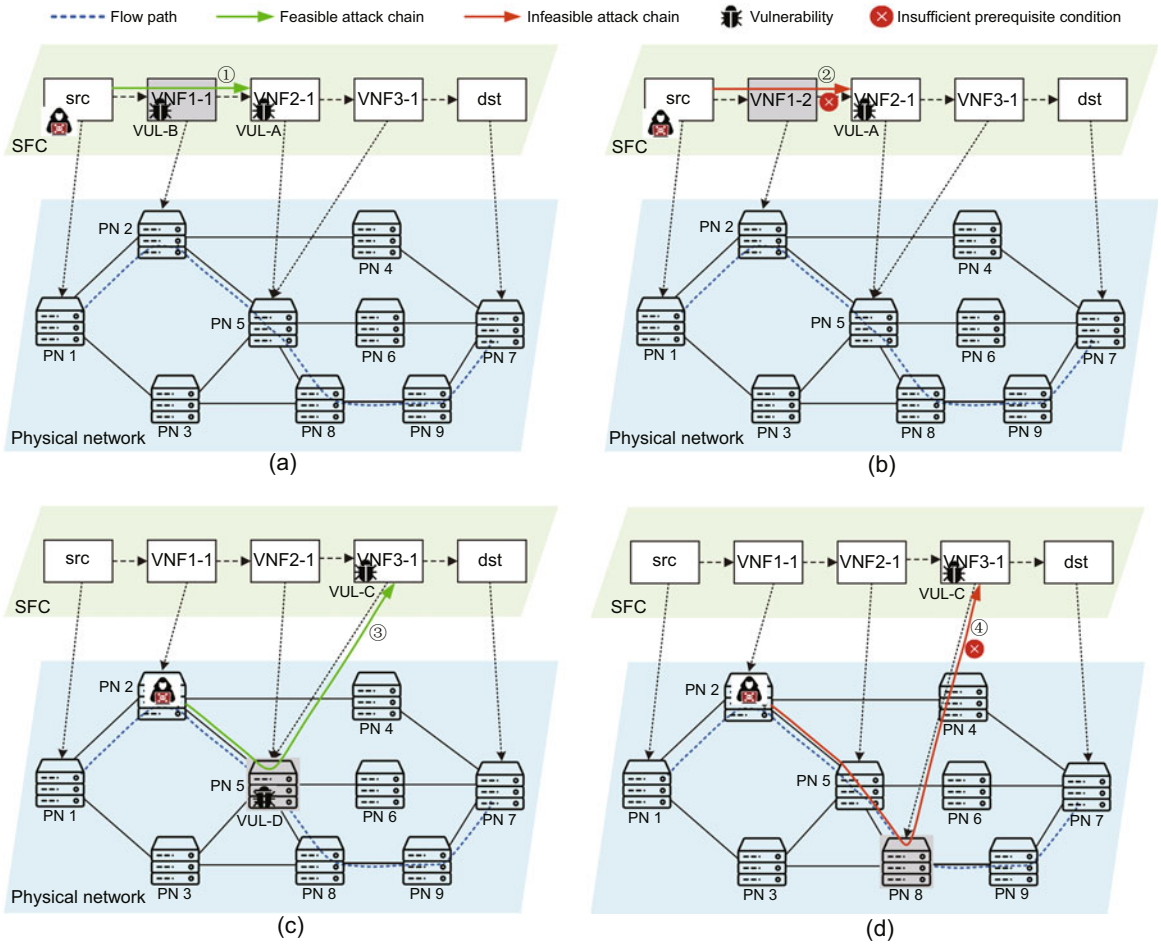

Fig. 1 The impact of vulnerability correlation on attack chains: (a) feasible attack chain in the virtual layer; (b) infeasible attack chain in the virtual layer; (c) feasible attack chain in the physical layer; (d) infeasible attack chain in the physical layer

firewall vulnerability VUL-B (e.g., CVE-2022-42475) to tamper with firewall rules, redirecting external port traffic to the intranet database port. This allows attackers to send a hypertext transfer protocol (HTTP) request containing malicious structured query language (SQL) statements to the database via database remote code execution vulnerability VUL-A (e.g., CVE-2021-34527), enabling arbitrary command execution and facilitating control of the database through a malicious request. Exploiting firewall vulnerability VUL-B redirects external traffic to the intranet database port, creating the conditions necessary to exploit VUL-A. However, if the equivalent functional firewall VNF1-2, which does not contain such vulnerabilities, replaces VNF1-1 in the SFC composition (as shown in Fig. 1b), the attack chain in the virtual layer is disrupted, preventing the exploitation of VUL-A. Therefore, the trustworthiness among different VNFs with equivalent functionality varies; that is, VNF2-1 trusts VNF1-2 more for the SFC composition. As shown in Fig. 1c, VNF3-1's exchange server is deployed on PN 5. Attackers can exploit the authentication bypass vulnerability VUL-D (e.g., CVE-2021-34473) to access PN 5's internal management interface. They can then leverage the privilege escalation vulnerability VUL-C (e.g., CVE-2021-31207) to elevate their privileges, ultimately stealing sensitive data from VNF3-1. The exploitation of VUL-D enables attackers to access the internal management interface, creating the conditions necessary for exploiting VUL-C. If PN 8 lacks the VUL-D and VNF3-1 is deployed on PN 8 (as shown in Fig. 1d), the conditions for exploiting VUL-C are not met, disrupting the attack chain in the physical layer. Thus, the trustworthiness between VNFs with equivalent functions and PNs differs. That is, VNF3-1 trusts PN 8 more for the SFC placement.

Vulnerability correlation and vulnerability differences between network entities lead to differences in trustworthiness among network entities, so different VNF combinations at the SFC composition stage and placement schemes at the SFC placement stage will result in different security effects of SFCs. However, existing security orchestration methods for SFC deployment (Torkzaban et al., 2019; Torkzaban and Baras, 2020; Zhang PY et al., 2021; Zhang Y et al., 2023; Zheng et al., 2023; Cao et al., 2024) ignore the impact of vulnerability correlation on trustworthiness among entities and consider only local trustworthiness optimization. These methods ignore both vulnerability correlation and the differences in vulnerabilities among network entities, thereby quantifying the security of cooperation among network entities as generic trust levels (or security levels). This makes it difficult to effectively disrupt the attack chains by security orchestration. Furthermore, these security orchestration methods on the basis of trustworthiness focus only on the VNF combination at the SFC composition stage or the PN selection at the SFC placement stage, failing to disrupt the attack chains in the virtual layer or the physical layer.

Therefore, we aim to globally determine the VNF combination and PN selection schemes based on the differentiated trustworthiness among network entities, to disrupt attack chains in SFCs and enhance their security. The major contributions of our article are as follows:

1. We consider, for the first time, the impact of vulnerability correlation and vulnerability differences among network entities on their differentiated trustworthiness, and propose a hierarchical trust model to dynamically assess trustworthiness among network entities.

2. Based on the assessed trustworthiness, we consider both virtual trust of the VNF combination at the SFC composition stage and physical trust of the PN selection at the SFC placement stage, and formulate the security-aware and cost-efficient SFC composition and placement (SCSCP) problem as an integer linear programming (ILP) model, which ensures that the attack chains in SFCs are disrupted as much as possible at a controllable cost.

3. A joint trust and cost global optimization (JTCGO) algorithm is proposed to obtain the SFC deployment solutions for SFC requests, where a trust update algorithm is used to update the trustworthiness among network entities and a VNF combination and PN selection (VCPS) algorithm globally finds the VNF combination schemes at the SFC composition stage and PN selection schemes at the SFC placement stage.

4. Finally, we carry out performance evaluations and comparisons and thereafter demonstrate the efficacy of our proposed algorithm. This algorithm can provide security-aware and cost-efficient solutions for SFC requests.

2 Related works

2.1 SFC security method

Anomaly detection (Hong et al., 2020; Wang WL et al., 2022, 2023; Tang et al., 2024) provides early warnings for SFC security by enabling NFV-management and network orchestration (NFV-MANO) to monitor infrastructure and VNFs. For example, Tang et al. (2024) proposed DTS-KD for time-series VNF status monitoring, while Wang WL et al. (2023) developed a federated learning-based framework allowing distributed slice managers to collaboratively train a global virtual machine (VM) anomaly model without sharing local metrics.

During attacks, moving target defense (MTD) (Peretz et al., 2020; Zhang T et al., 2023) enhances resilience via dynamic network changes and adaptive SFC migration. Virtual security functions (VSFs) further offer on-demand protection, with automation increasingly replacing manual configuration (Bagheri and Shameli-Sendi, 2023; Wang WL et al., 2024).

After successful attacks, fast recovery is critical. Reactive methods (Hu and Guo, 2021; Wang M et al., 2021) only respond after a fault occurs, incurring downtime; proactive methods (Peng et al., 2021; Cao et al., 2022; Alomari et al., 2023) predeploy backups to reduce recovery time at the cost of reserved resources.

These approaches mitigate attacks across prediction, resistance, and recovery stages, but have two limitations: They cannot prevent attacks at the source—they only detect or react—and resource-dependent strategies (e.g., backups and VSFs) incur overhead.

2.2 Security-aware SFC deployment method

To fundamentally reduce or even eliminate network attacks, it is crucial to disrupt the attack chains in SFCs at the service deployment stage. Existing works have quantified the security of network entities, representing it as a security level or trust to evaluate their security. Generally, the higher the security level or trust, the lower the likelihood of vulnerabilities being present or exploited, and the higher the security of the network entities.

Some works quantify the security of VNFs and select VNFs to SFCs that meet the security requirements of services, without considering the impact of PNs on security. In Zheng et al. (2023), the service functions with different security levels provided by various vendors were used to efficiently compose and embed the security-aware SFC to satisfy the security requirement, and an efficient algorithm called SCB-based deployment was proposed for the security-aware SFC deployment problem. Different from the pre-defined or pre-given security levels in prior work (Zheng et al., 2023), Zheng et al. (2024) identified how much security can be provided by a specific security-aware service function against a set of cyber attacks in advance to estimate the security level, and they effectively composed and embedded the services based on the estimated security level.

Some works (Torkzaban et al., 2019; Torkzaban and Baras, 2020; Zhang PY et al., 2021; Zhang Y et al., 2023; Cao et al., 2024) only quantify the security of PNs and deploy VNFs on the PNs that meet security requirements to ensure the security of services. For example, Cao et al. (2024) proposed Tail-ZeSec-6G to check all underlying general-purpose appliances and eliminate implicit trust issues when receiving service requests. After undergoing the security check and evaluation, Tail-ZeSec-6G will conduct a tailored resource allocation for service requests. Zhang Y et al. (2023) introduced a dynamic trust evaluation mechanism to evaluate the trust of each PN in the virtual network in real time and designed a deep reinforcement learning agent for resource allocation.

Few works consider the security of VNFs and PNs simultaneously. In Varadharajan et al. (2022), a trust model and property-based trust attestation mechanism were proposed to evaluate the trust of the VNFs that compose the network slice, which helps determine the trust of the VNFs, as well as the properties that should be satisfied by the virtual platforms.

In summary, the above-mentioned methods, in addition to focusing merely on the security of VNFs or PNs, ignore the impact of vulnerability correlation on attack chains. The current methods, when combining and deploying SFCs, rely only on general security levels or trust degrees, without considering the differences in trustworthiness caused by vulnerability correlation among the various network entities. Consequently, it is challenging to effectively disrupt the attack chains and ensure the security of SFCs.

3 Framework descriptions

The framework extends the trust plane and decision plane in the NFV/SDN-enabled network as depicted in Fig. 2.

In the trust plane, the trust management module is responsible for (1) collecting and storing the historical cooperation information reported by the network, and (2) managing the trust matrix related to the proposed hierarchical trust model. Historical collaboration information is the SFC status information reported from the data plane, which is represented in the form of the cooperation success rate of network entities. Based on the historical cooperation information, the hierarchical trust model is used to quantify the trustworthiness among the network entities in the trust update module, which comprises the virtual trust matrix and the physical trust matrix. The virtual trust matrix indicates the trustworthiness of VNFs to other VNFs, and the physical trust matrix indicates the trustworthiness of VNFs to PNs. The trust matrices are used in the VNF combination and PN selection module to obtain the SFC placement scheme. The hierarchical trust model is proposed in Section 4.2.

The decision plane consists of the trust update module and the VNF combination and PN selection module. The trust update module is responsible for periodically updating the trust matrix based on the historical collaboration information to obtain the latest trust, and the VNF combination and PN selection module is responsible for implementing the VNF combination and PN selection to obtain SFC deployment solutions based on the trust matrix and the physical network. For this purpose, a trust update algorithm and the VCPS algorithm are proposed in Sections 5.1 and 5.2, respectively

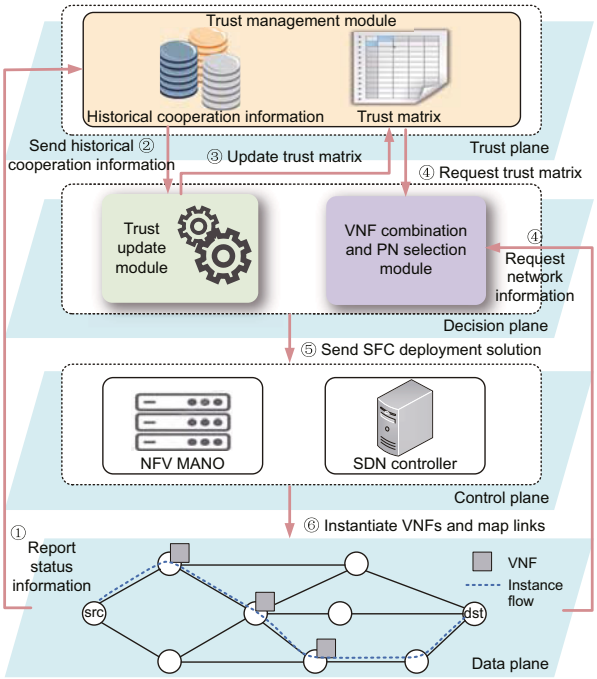

Fig. 2 Framework and processing procedure

Based on the obtained SFC deployment solutions, NFV-MANO is responsible for deploying and instantiating VNFs in the physical network. SDN controller takes on the task of managing the traffic between VNF instances, ensuring that the traffic is routed along specified paths. The physical network in the data layer provides the necessary resources and supportive infrastructure for VNF instances to deliver services.

4 Problem model

In this section, we first define the symbols and introduce the hierarchical trust model. Then, the trustworthiness of the SFC is given and the SCSCP problem is formulated as an ILP model.

4.1 Symbol definition

The physical network is denoted as $G = (N, E)$, where N and E denote the sets of PNs and physical links (PLs) interconnecting these PNs, respectively. Each PN $n_i \in N$ has a finite amount of available central processing unit (CPU) resource capacity $C^{\text{cpu}}_{n_i}$. Each PL $(n_i, n_j) \in E$ interconnecting PN n_i and n_j has a finite amount of available bandwidth resource capacity $C^{\text{bw}}_{n_i n_j}$ and a certain transmission delay $d_{n_i n_j}$. Due to the differences between PNs and PLs in infrastructure types, operation and maintenance, the required cost for consuming unit CPU and unit bandwidth resources is different, which is represented by $c^{\text{cpu}}_{n_i}$ and $c^{\text{bw}}_{n_i n_j}$, respectively.

In the network, each type of VNF with equivalent functionality can be provided by various service vendors, and the NFV orchestrator can select the optimal VNF from each type of VNF that meets the functionality to compose the SFCs. $F = \{F_1, F_2, \ldots, F_{|F|}\}$ is used to denote the candidate VNF set, which includes a total of $|F|$ types of VNFs. Each type of VNF consists of multiple VNFs with equivalent functionality, which can be denoted as

$F_k = \{f_{k1}, f_{k2}, \ldots, f_{k|F_k|}\}$, and the number of VNFs in set F_k is $|F_k|$. All VNFs $f_{ki} \in F_k$ can provide the equivalent functionality but with different providers, so there are disparities in vulnerabilities and resource requirements. The CPU requirement of VNF f_{ki} is denoted as $\text{cpu}_{f_{ki}}$.

The SFC request $s \in S$ is denoted as $s = \{F^s, L^s, \text{src}^s, \text{dst}^s, B^s, D^s\}$. The SFC request consists of multiple VNFs, which can be denoted as $F^s = \left\{f_1^s, f_2^s, \ldots, f_{|F^s|}^s\right\}$, and the set of virtual links that interconnect VNFs from the source node src^s to the destination node dst^s in an ordered manner is denoted as L^s. The bandwidth requirement and maximum delay constraint are denoted as B^s and D^s respectively. The SFC request focuses only on the functionalities of the requested VNFs, and does not specify the concrete VNFs composing the SFC. In other words, it only specifies the set F_k to which the VNF f_i^s belongs, but it does not specify which VNF f_{ki}. The aim of our work is to determine which VNF f_{ki} to select for composing the SFC to disrupt the attack chains.

To indicate the solutions of the SFC request s, we define the following binary variables:

1. The binary variable $\varepsilon_{f_i^s}^{F_k}$ is used to represent whether the requested VNF f_i^s should be selected from the equivalent VNF set F_k or not to meet the required functionality, which can be expressed as follows:

$$\varepsilon_{f_i^s}^{F_k} = \begin{cases} 1, & \text{if VNF } f_i^s \text{ should be selected from } F_k, \\ & \text{i.e., } f_i^s \in F_k, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

2. The binary variable $\delta_{f_i^s}^{f_{ki}}$ is used to represent whether the VNF $f_{ki} \in F_k$ is selected for the request VNF f_i^s or not, that is $\varepsilon_{f_i^s}^{F_k} = 1$ and $f_i^s = f_{ki}$, which can be expressed as follows:

$$\delta_{f_i^s}^{f_{ki}} = \begin{cases} 1, & \text{if VNF } f_{ki} \text{ is selected for the} \\ & \text{requested VNF } f_i^s, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

3. The binary variable $\alpha_{f_i^s}^{n_i}$ is used to represent whether the VNF f_i^s is placed on PN n_i or not, which can be expressed as follows:

$$\alpha_{f_i^s}^{n_i} = \begin{cases} 1, & \text{if VNF } f_i^s \text{ is placed on PN } n_i, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

4. The binary variable $\beta_{f_i^s f_j^s}^{n_i n_j}$ is used to represent whether the virtual link $\left(f_i^s, f_j^s\right)$ is mapped in PL (n_i, n_j) or not, which can be expressed as follows:

$$\beta_{f_i^s f_j^s}^{n_i n_j} = \begin{cases} 1, & \text{if virtual link } \left(f_i^s, f_j^s\right) \text{ is mapped in} \\ & \text{physical link } (n_i, n_j), \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

4.2 Hierarchical trust model

To disrupt the attack chains from the connected VNFs and placed PNs as much as possible, the hierarchical trust model is proposed to assess the trustworthiness among entities. Hierarchical trust model consists of virtual trust and physical trust, where virtual trust represents the trustworthiness among VNFs and physical trust represents the trustworthiness of VNFs to their placed PNs. Both virtual trust and physical trust consist of the direct trust and indirect trust (Jorquera Valero et al., 2023).

Direct trust is a crucial concept that can be assessed, representing the trust to the target entities. In the network, we can quantify the historical cooperation information among entities (including VNFs and PNs) to define and calculate the direct trust. The historical cooperation information documents the cooperation situations of entities when they form the SFCs to provide services under attacks. It directly reflects whether the vulnerabilities in entities have been exploited by attackers to form the attack chains, which can represent the likelihood of the existence of attack chains among the two entities. The historical cooperation information between entities v_i and v_j at time t is represented as the set of time-series probability data:

$$\lambda_{v_i v_j}(t) = \left\{\lambda_{v_i v_j}^1, \lambda_{v_i v_j}^2, \ldots, \lambda_{v_i v_j}^t\right\}, \tag{5}$$

where v_i denotes a VNF f_{ki}, v_j denotes a VNF f_{gj} or PN n_i, and $\lambda_{v_i v_j}^{t_i}$ denotes the success rate of cooperation services between entities v_i and v_j at time t_i, serving as the basis for assessing direct trust. It should be noted that the effectiveness of historical cooperation information decreases over time because VNFs and PNs are constantly upgraded to patch flaws such as vulnerabilities. Early cooperative behavior may no longer accurately reflect the current direct trust. Exponential decay model prioritizes recent data with rapid responsiveness to changes and

does not overly emphasize long-term historical data, which makes it well suited for computing direct trust based on historical cooperation information. So the exponential decay model is used for the time decay factor to reduce the effect of time on direct trust:

$$\varepsilon_{t_i}(t)=\mathrm{e}^{-\mu(t-t_i)}, \tag{6}$$

where $\varepsilon_{t_i}(t)$ denotes the decay factor of historical cooperative record at time t_i. Therefore, direct trust of entities v_i to v_j can be obtained by

$$\mathrm{DT}_{v_iv_j}(t)=\frac{\sum\limits_{t_i\in(1,t)}\varepsilon_{t_i}(t)\,\lambda_{v_iv_j}^{t_i}}{\sum\limits_{t_i\in(1,t)}\varepsilon_{t_i}(t)}. \tag{7}$$

Here, the larger the value of $\mathrm{DT}_{v_iv_j}(t)$, the lower the likelihood of an attack chain from v_j to v_i, and the more secure the cooperation service of v_i and v_j.

Indirect trust involves indirectly assessing the trust to a target entity through the direct trust that third-party VNFs have to the target entity. This enables a comprehensive assessment of the trustworthiness to the target entity. Attackers exploit vulnerabilities to attack entities, and there exists a correlation among these vulnerabilities. Given that the vulnerability profiles of different entities vary, the direct trust held by different third-party VNFs to the target entity exerts varying impacts on the indirect trust to the target entity. So heterogeneity can measure the difference between entities, and thus is used to represent the importance of the direct trust of the third-party VNF to the target entity in the indirect trust assessment. In a previous work (Zhang QQ et al., 2021), heterogeneity was defined by the number of common vulnerabilities and differential vulnerabilities in entities. Because software upgrading, vulnerability patching, and similar activities will have an impact on the number of vulnerabilities of VNFs, heterogeneity also changes over time. So heterogeneity $\phi_{v_iv_k}(t)$ between entities v_i and v_k can be expressed as follows:

$$\phi_{v_iv_k}(t)=1-\frac{2\varphi_{v_iv_k}(t)}{\varphi_{v_i}(t)+\varphi_{v_k}(t)}, \tag{8}$$

where $\varphi_{v_i}(t)$ and $\varphi_{v_k}(t)$ denote the numbers of total vulnerabilities in v_i and v_k at time t respectively, and $\varphi_{v_iv_k}$ denotes the number of common vulnerabilities causing consistent processing results between v_i and v_k at time t. So $1-\phi_{v_iv_k}(t)$ is used to represent the importance of the direct trust of the third-party VNF v_k to the target entity in the indirect trust assessment. The less heterogeneity with the third-party VNF v_k, the more likely it is to have the same vulnerabilities, and the more important the direct trust $\mathrm{DT}_{v_kv_j}(t)$ of the third party VNF v_k to the target entity v_j is. Therefore, indirect trust of entity v_i to v_j can be obtained by

$$\mathrm{IT}_{v_iv_j}(t)=\frac{\sum\limits_{v_k\in F}(1-\phi_{v_iv_k}(t))\cdot\mathrm{DT}_{v_kv_j}(t)}{\sum\limits_{v_k\in F}\left\lceil\mathrm{DT}_{v_kv_j}(t)\right\rceil\cdot(1-\phi_{v_iv_k}(t))}, \tag{9}$$

where F denotes the VNF set F. $\left\lceil\mathrm{DT}_{v_kv_j}(t)\right\rceil$ means that direct trust is rounded up; that is, only the third-party VNFs that have the direct cooperation information with the target entities are considered.

Therefore, trustworthiness among any entities $T_{v_iv_j}(t)$ can be obtained by the weighted sum of direct trust and indirect trust, which can be expressed as follows:

$$T_{v_iv_j}(t)=w_1\cdot\mathrm{DT}_{v_iv_j}(t)+(1-w_1)\cdot\mathrm{IT}_{v_iv_j}(t), \tag{10}$$

where w_1 denotes the direct trust weight and $1-w_1$ denotes the indirect trust weight. Although it does not accurately represent the probability of the existence of attack chains, the value of trustworthiness indicates how likely it is that attack chains exist. The higher the trustworthiness $T_{v_iv_j}(t)$, the lower the likelihood of an attack chain from v_j to v_i.

Based on the trust assessment, we can obtain the trustworthiness among all VNFs and trustworthiness of all VNFs to all PNs at time t, which are denoted as the virtual trust matrix and the physical trust matrix, respectively. Virtual trust matrix is a $\sum\limits_{F_k\in F}|F_k|\times\sum\limits_{F_k\in F}|F_k|$ matrix and $\sum\limits_{F_k\in F}|F_k|$ represents the total number of all VNFs in F. When $f_{ki}=f_{kj}$, $T_{f_{ki}f_{kj}}(t)=1$, and when f_{ki} and f_{gj} are the VNFs with the equivalent functionality, that is, $k=g$ and $i\neq j$, $T_{f_{ki}f_{kj}}(t)=0$. This is because we do not consider the scenario of backup requirements, so there will be no cooperative information among VNFs with equivalent functionality. Physical trust matrix is a $\sum\limits_{F_k\in F}|F_k|\times|N|$ matrix and $|N|$ is the number of PNs.

4.3 SFC trustworthiness

Trustworthiness of the requested SFC consists of virtual trust and physical trust. Virtual trust of

the requested SFC is determined by the VNF combination at the SFC composition stage, which can be expressed as follows:

$$\begin{aligned}\mathrm{VT}^s(t) &= \prod_{f_i^s\in F^s}\sum_{F_k\in F}\sum_{f_{ki}\in F_k}\sum_{F_g\in F}\sum_{f_{gi}\in F_g}\mathcal{F}(t),\\ \mathcal{F}(t) &= \varepsilon_{f_i^s}^{F_k}\delta_{f_i^s}^{f_{ki}}\varepsilon_{f_{i+1}^s}^{F_g}\delta_{f_{i+1}^s}^{f_{gi}}T_{f_i^s f_{i+1}^s}(t).\end{aligned} \tag{11}$$

Physical trust of the requested SFC is determined by the placed PNs of the selected VNFs at the SFC placement stage, which can be expressed as follows:

$$\mathrm{PT}^s(t)=\prod_{f_i^s\in F^s}\sum_{F_k\in F}\sum_{f_{ki}\in F_k}\sum_{n_i\in N}\varepsilon_{f_i^s}^{F_k}\delta_{f_i^s}^{f_{ki}}\alpha_{f_i^s}^{n_i}T_{f_{ki}n_i}(t). \tag{12}$$

Therefore, trustworthiness of the requested SFC at time t can be obtained by the weighted sum of virtual trust and physical trust, which can be expressed as follows:

$$\mathrm{Trust}^s = w_2\cdot\mathrm{VT}^s(t)+(1-w_2)\cdot\mathrm{PT}^s(t), \tag{13}$$

where w_2 and $1-w_2$ denote the virtual trust weight and physical trust weight, respectively. The larger the trust, the lower the likelihood of attack chains, and the more secure the requested SFC. The higher the trustworthiness of SFC Trust^s, the lower the likelihood of attack chains.

Fig. 3 shows the four solutions of different VNF combinations and PN selections for the same SFC request, and Table 1 lists the corresponding trust values of these solutions when the virtual trust weight w_2 is 0.5. We can observe that VNF combination solutions and PN selection solutions will affect the trustworthiness of SFCs. Moreover, the solutions with greater virtual trust or physical trust may not necessarily gain a higher overall trust. Consequently, Solution D shown in Fig. 3d is the best among the four solutions, solely from the perspective of trustworthiness. Thus, it is imperative to consider the SFC composition and placement from a global perspective, to disrupt the attack chains in SFCs as much as possible.

4.4 SCSCP problem

In practical scenarios, it is unrealistic to simply pursue the trustworthiness of SFCs while ignoring the cost. Therefore, we take the maximum unit-cost trust Trust2Cost as the optimization objective of the SCSCP problem, which fully uses cost. According to the hierarchical trust model, the trust of SFCs can be obtained using Eq. (13). Cost consists of the CPU cost towards consumption by VNFs and the bandwidth cost against consumption by virtual links, which can be obtained as follows:

$$\begin{aligned}\mathrm{Cost}^s=&\sum_{f_i^s\in F^s}\sum_{F_k\in F}\sum_{f_{ki}\in F_k}\sum_{n_i\in N}\varepsilon_{f_i^s}^{F_k}\delta_{f_i^s}^{f_{ki}}\alpha_{f_i^s}^{n_i}\mathrm{cpu}_{f_i^s}c_{n_i}^{\mathrm{cpu}}\\ &+\sum_{(f_i^s,f_j^s)\in L^s}\sum_{(n_i,n_j)\in E}\beta_{f_i^s f_j^s}^{n_i n_j}B^s c_{n_i n_j}^{\mathrm{bw}}.\end{aligned} \tag{14}$$

Therefore, unit-cost trust Trust2Cost can be obtained using the following equation:

$$\mathrm{Trust2Cost}=\frac{\mathrm{Trust}^s}{\mathrm{Cost}^s}. \tag{15}$$

To ensure that the solution meets the requirements of the SFC request, the SCSCP problem should meet the following constraints:

1. Functionality constraint. Any requested VNF $f_i^s\in F^s$ must be selected to form the equivalent VNF set that meets the required functionality, and only one of the VNFs in F^s can be selected as the request VNF f_i^s. This constraint ensures that the selected VNFs can meet the functionality of the SFC request at the SFC composition stage, which can be expressed as follows:

$$\begin{aligned}&\forall s\in S,\ \forall f_p^s\in F^s,\ \exists F_k\in F,\\ &\text{if } \varepsilon_{f_p^s}^{F_k}=1,\ \sum_{f_{ki}\in F_k}\delta_{f_p^s}^{f_{ki}}=1,\end{aligned} \tag{16}$$

where f_p^s denotes the p^{th} VNF in request s.

2. Flow constraint. For each PN n_i, the numbers of all incoming and outgoing flows of the mapped virtual links should be equal when placing any SFC request, which can be expressed as follows:

$$\begin{aligned}&\forall n_i\in N,\ \forall s\in S,\\ &\sum_{(f_i^s,f_j^s)\in L^s}\sum_{n_j\in N}\beta_{f_i^s f_j^s}^{n_i n_j}+Z_1\\ =&\sum_{(f_i^s,f_j^s)\in L^s}\sum_{n_j\in N}\beta_{f_i^s f_j^s}^{n_j n_i}+Z_2.\end{aligned} \tag{17}$$

In Eq. (17), if $\alpha_{\mathrm{src}^s}^{n_i}=1$, Z_1=1; otherwise, Z_1=0. If $\alpha_{\mathrm{dst}^s}^{n_i}=1$, Z_2=1; otherwise, Z_2=0.

3. Loop constraint. The loop constraint ensures that the mapped PLs of all virtual links $(f_i^s,f_j^s)\in$

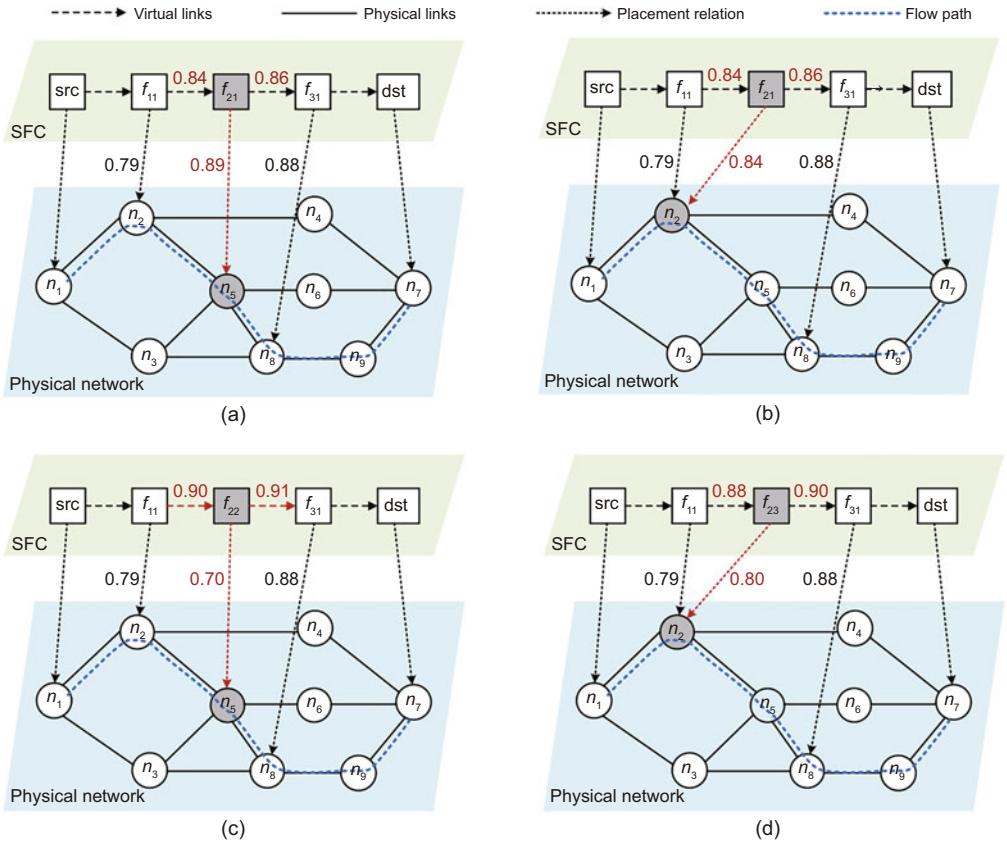

Fig. 3 Solutions with different schemes for VNF f_2^s: (a) Solution A (f_{21} is selected as f_2^s and placed on PN n_5); (b) Solution B (f_{21} is selected as f_2^s and placed on PN n_2); (c) Solution C (f_{22} is selected as f_2^s and placed on PN n_5); (d) Solution D (f_{23} is selected as f_2^s and placed on PN n_2). The comparison between Solutions A and B illustrates the trust difference of different VNFs to the same PN and the impact of SFC placement on trust, and the comparison between Solutions A and C illustrates the impact of SFC composition on trust. Trust of Solution D is larger than that of Solution A which has larger physical trust, and that of Solution C which has larger virtual trust

Table 1 Service trust of different solutions in Fig. 3

Solution	Virtual trust	Physical trust	Overall trust
A	$0.84 \times 0.86 = 0.7224$	$0.79 \times 0.89 \times 0.88 =$ **0.618 728**	$0.5 \times (0.7224 + 0.618\ 728) = 0.670\ 564$
B	$0.84 \times 0.86 = 0.7224$	$0.79 \times 0.84 \times 0.88 = 0.583\ 968$	$0.5 \times (0.7224 + 0.583\ 968) = 0.653\ 184$
C	$0.90 \times 0.91 =$ **0.8190**	$0.79 \times 0.70 \times 0.88 = 0.486\ 640$	$0.5 \times (0.8190 + 0.486\ 640) = 0.652\ 820$
D	$0.88 \times 0.90 = 0.7920$	$0.79 \times 0.80 \times 0.88 = 0.556\ 160$	$0.5 \times (0.7920 + 0.556\ 160) =$ **0.674 080**

Best results are in bold

L^s do not form the route loops, which should be guaranteed as follows:

$$\forall s \in S,\ \forall \left(f_i^s, f_j^s\right) \in L^s,\ \forall n_i \in N,$$
$$\sum_{n_j \in N} \beta_{f_i^s f_j^s}^{n_i n_j} \le 1 \text{ and } \sum_{n_j \in N} \beta_{f_i^s f_j^s}^{n_j n_i} \le 1. \tag{18}$$

4. Order constraint. To ensure the execution of VNFs in sequential order, a constraint is needed to ensure that the requested VNFs of a flow are created along that path from source PN to destination PN. If f_i^s is placed on the PN n_i, the virtual link (f_i^s, f_{i+1}^s) must be mapped into one PL with n_i as its source node. So the order constraint can be expressed as follows:

$$\forall s \in S,\ \forall f_i^s \in F^s,$$
$$\text{if } \alpha_{f_i^s}^{n_i} = 1,\ \sum_{n_j \in N} \beta_{f_j^s f_{j+1}^s}^{n_i n_j} = 1. \tag{19}$$

5. Delay constraint. The SFC delay should be

less than the maximum delay constraint D^s, and we only consider the transmission delay, which can be expressed as follows:

$$\forall s \in S, \sum_{\left(f_i^s, f_j^s\right) \in L^s} \sum_{(n_i, n_j) \in E} \beta_{f_i^s f_j^s}^{n_i n_j} d_{n_i n_j} \leq D^s. \quad (20)$$

6. Resource constraint. The consumed resource requirements of SFC placement into the physical network should not exceed the available resource capacity of PNs and PLs, which can be expressed respectively as follows:

$$\forall n_i \in N, \sum_{s \in S} \sum_{f_i^s \in F^s} \sum_{F_k \in F} \sum_{f_{ki} \in F_k} \varepsilon_{f_i^s}^{F_k} \delta_{f_i^s}^{f_{ki}} \alpha_{f_i^s}^{n_i} \mathrm{cpu}_{f_i^s} \leq C_{n_i}^{\mathrm{cpu}}, \quad (21)$$

$$\forall (n_i, n_j) \in E, \sum_{\left(f_i^s, f_j^s\right) \in L^s} \beta_{f_i^s f_j^s}^{n_i n_j} B^s \leq C_{n_i n_j}^{\mathrm{bw}}. \quad (22)$$

7. Placement constraint. Each request VNF can be deployed only once on PNs, and each virtual link can be mapped to a PL only once, which can be expressed respectively as follows:

$$\forall f_i^s \in F^s, \sum_{n_i \in N} \alpha_{f_i^s}^{n_i} = 1, \quad (23)$$

$$\forall \left(f_i^s, f_j^s\right) \in L^s, \sum_{(n_i, n_j) \in E} \beta_{f_i^s f_j^s}^{n_i n_j} \geq 1. \quad (24)$$

Therefore, the SCSCP problem can be formulated as an ILP model:

$$\begin{aligned} &\max \ \mathrm{Trust2Cost} \\ &\text{s.t. constraints (16) } - \text{(24) are satisfied.} \end{aligned} \quad (25)$$

5 JTCGO algorithm

To provide solutions for SFC requests, ensure service performance, and prevent attacks, we propose the JTCGO algorithm, which encompasses a trust update algorithm and a VCPS algorithm. As presented in Algorithm 1, the trust update algorithm periodically updates the trust matrix, eliminating the impacts caused by software upgrades and vulnerability patching (line 3 in Algorithm 1). At the same time, once the SFC request arrives, the VCPS algorithm obtains the SFC deployment solutions for SFC requests based on the latest trust matrix, including VNF combination schemes and PN selection schemes (lines 4–8 in Algorithm 1). As illustrated in Fig. 3, merely pursuing maximum virtual trust or maximum physical trust makes it difficult to guarantee the performance of the solutions. This renders the two-stage approach of first SFC composition and then SFC placement infeasible. Therefore, the VCPS algorithm is proposed to solve VNF combinations at the SFC composition stage and PN selection at the SFC placement stage in a holistic manner.

Algorithm 1 JTCGO algorithm

1: **Input:** physical network G, SFC request set S, VNF set F, heterogeneity $\phi_{f_{ki}f_{gj}}(t)$, PN set N, historical cooperation information $\lambda_{v_i v_j}(t)$, weights w_1 and w_2, and algorithm parameters
2: **Output:** the set of solutions Solution
// Update trust matrix
3: Periodically update trust matrices:
$\boldsymbol{M}_{\mathrm{VT}}(t)$, $\boldsymbol{M}_{\mathrm{PT}}(t)$=Trust Update Algorithm
// Obtain the solutions
4: **while** $S \neq \varnothing$ **do**
5: Obtain $\lambda_{v_i v_j}(t)$ and $\phi_{f_{ki}f_{gj}}$ at current time t
6: Solution=VCPS Algorithm
7: Solution.append(solution)
8: **end while**
9: **return** Solution

5.1 Trust update algorithm

The trust update algorithm updates the virtual trust matrix and the physical trust matrix in accordance with the historical cooperation information and heterogeneity at the current time t, as shown in Algorithm 2. At the first step of the algorithm, it acquires the historical cooperation information at the current time t, which includes the historical cooperation information among VNFs as well as that between VNFs and PNs (line 3 in Algorithm 2). Based on this historical cooperation information, direct trust $\mathrm{DT}_{f_{ki}f_{gj}}(t)$ and $\mathrm{DT}_{f_{ki}n_i}(t)$ at time t can be directly computed via Eq. (7) (lines 4–11 in Algorithm 2). Subsequently, relying on the latest direct trust and heterogeneity, indirect trust $\mathrm{IT}_{f_{ki}f_{gj}}(t)$ and $\mathrm{IT}_{f_{ki}n_i}(t)$ at time t can be calculated through Eq. (9) (lines 12–27 in Algorithm 2). Then, trust $T_{f_{ki}f_{gj}}(t)$ and $T_{f_{ki}n_i}(t)$ at time t can be obtained by taking the weighted sum of direct trust and indirect trust (lines 19 and 23 in Algorithm 2). Finally, virtual trust matrix $\boldsymbol{M}_{\mathrm{VT}}(t)$ and physical trust matrix $\boldsymbol{M}_{\mathrm{PT}}(t)$ are updated separately (line 28 in Algorithm 2). These trust matrices will serve as the input of the VCPS algorithm for solutions.

Algorithm 2 Trust update algorithm

Input: VNF set F, PN set N, historical cooperation information $\lambda_{v_i v_j}(t)$, heterogeneity $\phi_{f_{ki} f_{gj}}(t)$, and direct trust weight w_1
Output: updated trust matrices $\boldsymbol{M}_{\mathrm{VT}}(t)$ and $\boldsymbol{M}_{\mathrm{PT}}(t)$
Obtain the latest cooperation information at time t
// Calculate direct trust
for $f_{ki} \in F$ **do**
  for $f_{gj} \in F$ **do**
    Calculate $\mathrm{DT}_{f_{ki} f_{gj}}(t)$ using Eq. (7)
  end for
  for $n_i \in N$ **do**
    Calculate $\mathrm{DT}_{f_{ki} n_i}(t)$ using Eq. (7)
  end for
end for
// Calculate indirect trust
for $f_{ki} \in F$ **do**
  for $f_{gj} \in F$ **do**
    if $k = g$ and $i \neq j$ **then**
      $\mathrm{IT}_{f_{ki} f_{gj}}(t) = 0$
    else if $k = g$ and $i = j$ **then**
      $\mathrm{IT}_{f_{ki} f_{gj}}(t) = 1$
    else
      Calculate $\mathrm{IT}_{f_{ki} f_{gj}}(t)$ using Eq. (9)
    end if
    // Calculate virtual trust
    $T_{f_{ki} f_{gj}}(t) = w_1 \cdot \mathrm{DT}_{f_{ki} f_{gj}}(t) + (1 - w_1) \cdot \mathrm{IT}_{f_{ki} f_{gj}}(t)$
  end for
  // Calculate physical trust
  for $n_i \in N$ **do**
    Calculate $\mathrm{IT}_{f_{ki} n_i}(t)$ using Eq. (9)
    $T_{f_{ki} n_i}(t) = w_1 \cdot \mathrm{DT}_{f_{ki} n_i}(t) + (1 - w_1) \cdot \mathrm{IT}_{f_{ki} n_i}(t)$
  end for
end for
Update $\boldsymbol{M}_{\mathrm{VT}}(t)$ and $\boldsymbol{M}_{\mathrm{PT}}(t)$
return $\boldsymbol{M}_{\mathrm{VT}}(t)$ and $\boldsymbol{M}_{\mathrm{PT}}(t)$

5.2 VCPS algorithm

The VCPS algorithm is ingeniously proposed based on the neural population dynamics optimization algorithm (NPDOA) with the aim of effectively solving the SCSCP problem. The VCPS algorithm is required to integrally perform the VNF combination and PN selection process, thus taking both virtual trust and physical trust into consideration.

NPDOA (Ji et al., 2024) is an optimization algorithm for addressing NP-hard problems through its biology-inspired computational framework. NPDOA treats each candidate solution as a neural population state, and uses parallel multi-population exploration of the solution space to reduce computational complexity. Additionally, NPDOA effectively balances the exploration and exploitation by attractor trending, coupling disturbance, and information projection strategies. This endows NPDOA with the advantages in terms of rapid convergence capability, global optimization performance, and stable solution quality. Therefore, by combining the unique characteristics of the SCSCP problem with the advantages of NPDOA, the VCPS algorithm using NPDOA is proposed to effectively address the SCSCP problem.

The VCPS algorithm solves the SCSCP problem by simulating the dynamics of multiple neural populations with the same number of neurons when the algorithm performs cognitive activities. For an SFC request s, a solution to the problem $X_i = (x_1, x_2, \ldots, x_D)$ is considered the neural state of a neural population with D neurons, where D is equal to $2|F^s|$, and $|F^s|$ denotes the number of the requested VNFs in the SFC request s. The former $|F^s|$ neurons represent the selected VNFs for the requested VNFs, and the state space of these neurons contains integers from 1 to $|F_k|$, where $|F_k|$ denotes the size of the VNF set with requested functionality. The latter $|F^s|$ neurons represent the placed PNs for the requested VNFs, and the state space of these neurons contains integers from 1 to $|N|$, where $|N|$ denotes the number of PNs in the network. The VNF combination and PN selection jointly form the neural state of a neural population, and integrated optimization is achieved through the proposed algorithm. The dynamics of neural populations can be calculated using Eq. (15), which represents the optimization objective value Trust2Cost.

At the beginning of the optimization process, the neural states of several neural populations $P = \{X_1, X_2, \ldots, X_{n_\mathrm{p}}\}$ are randomly generated in the neural state space, and n_p is the number of neural populations. Then, the neural states of these neural populations change according to the following three neural population dynamics strategies.

1. Attractor trending strategy: The attractor trending strategy steers neural populations toward optimal decisions, thereby guaranteeing the exploitation capability. In the neural state space, the neural states of neural populations tend toward multiple attractors. As such, the neural states with optimal dynamics are regarded as attractors, while the neural states of the remaining neural populations randomly

converge to one of these attractors. The number of attractors is set to $\lfloor a \cdot n_{\mathrm{p}} \rfloor$, where $\lfloor \cdot \rfloor$ refers to the operation of rounding a number down to the greatest integer less than or equal to the number, and $a \in (0,1)$ represents the proportion of attractors. Thus, the attractor set can be denoted as $\mathrm{ATT} = \{\mathbf{att}_1, \mathbf{att}_2, \ldots, \mathbf{att}_{\lfloor a \cdot n_{\mathrm{p}} \rfloor}\}$. The linear dynamical system (Vyas et al., 2020), which serves as a model of neural population dynamics, is used to simulate the attractor trending behavior of neural populations. The formula used is as follows:

$$\begin{aligned} &\boldsymbol{X}_{\mathrm{attract},i} = f_{\mathrm{attract}} \cdot r_{\mathrm{att}}^2 \cdot (\mathbf{att}_k - \boldsymbol{X}_i) - \boldsymbol{\theta}_i, \\ &\boldsymbol{\theta}_i \sim N(\mathbf{0}, \boldsymbol{U} - \boldsymbol{L}), \end{aligned} \tag{26}$$

where f_{attract} is the scaling factor of the attractor trending, r_{att} is a random number in $[0,1]$, $\mathbf{att}_k$ is the k^{th} attractor in ATT, k is a random value in $[1, \lfloor a \cdot n_{\mathrm{p}} \rfloor]$, $\boldsymbol{\theta}_i$ is the zero-mean Gaussian noise vector of the i^{th} neural state with covariance $\boldsymbol{U} - \boldsymbol{L}$, $\boldsymbol{U} = (U_1, U_2, \ldots, U_D)$ represents the peak values of all the neural states, and $\boldsymbol{L} = (L_1, L_2, \ldots, L_D)$.

Eq. (26) indicates that the neural state of a neural population tends to approximate that of a random attractor. Gaussian noise is incorporated into the dynamics formula, which serves to model the interference originating from external inputs (Valente et al., 2022). Moreover, the attractor-functioning neural states remain unaffected by the attractor trending approach. This search strategy based on the attractor trending behavior ensures the exploitation ability of the algorithm.

2. Coupling disturbance strategy: The coupling disturbance strategy deviates neural populations from attractors by coupling with other neural populations, thus improving exploration ability. The neural state of a particular neural population is also influenced by the neural states of other neural populations, and the interactions between neural populations occur through additive or diffusive coupling. In additive coupling, the input to a neural population is the average of the sum of all neural states. The formula used is as follows:

$$\boldsymbol{X}_{\mathrm{add},i} = r_2 \cdot \frac{1}{n_{\mathrm{p}}} \sum_{j \in [1, n_{\mathrm{p}}]} \boldsymbol{X}_j, \tag{27}$$

where r_2 is a random number in $[0, 0.5]$. In diffusive coupling, the input to a neural population is a function of the sum of the differences between its neural state and other neural states. The formula used is as follows:

$$\boldsymbol{X}_{\mathrm{dif},i} = r_3 \cdot \frac{1}{n_{\mathrm{p}}} \sum_{j \in [1, n_{\mathrm{p}}]} (\boldsymbol{X}_i - \boldsymbol{X}_j), \tag{28}$$

where r_3 is a random number in $[0, 0.5]$. The proposed algorithm uses a combination of two couplings to simulate the effects of other neural populations. The formula used is as follows:

$$\boldsymbol{X}_{\mathrm{couple},i} = f_{\mathrm{couple}} \cdot (\boldsymbol{X}_{\mathrm{add},i} + \boldsymbol{X}_{\mathrm{dif},i}), \tag{29}$$

where f_{couple} is the scaling factor of the coupling disturbance.

Eq. (29) shows that one neural population is influenced by another through two types of coupling. The disturbance from other neural populations leads the neural population to deviate from the influence of the attractors. This search strategy, which is founded on the coupling between neural populations, can contribute to boosting the exploration capabilities of the algorithm.

3. Information projection strategy: The information projection strategy controls the communication between neural populations, enabling a transition from exploration to exploitation. Information interchange and interaction among neural populations occur through the communication subspace. This subspace selectively captures the traits of a specific neural population, and then transfers them to another neural population (Semedo et al., 2019). In our proposed algorithm, the matrices denoting adjacency and communication intensity are used to represent the communication subspace for information projection between neural populations.

In the attractor trending, information transfer between an attractor and another neural state of a neural population is achieved by a $1 \times D$ adjacency matrix $\boldsymbol{C}_{\mathrm{attract},i}$ with binary random values and a $1 \times D$ communication intensity matrix $\boldsymbol{R}_{\mathrm{attract},i}$ with randomly generated values. The formula used is as follows:

$$\boldsymbol{X}_{\mathrm{C\_attract},i} = \boldsymbol{C}_{\mathrm{attract},i} \circ \boldsymbol{R}_{\mathrm{attract},i} \circ \boldsymbol{X}_{\mathrm{attract},i}, \tag{30}$$

where $\circ$ denotes the Hadamard product.

In the coupling disturbance, information transfer between neural populations is also used by another $1 \times D$ adjacency matrix $\boldsymbol{C}_{\mathrm{couple},i}$ with binary random values and a $1 \times D$ communication intensity matrix $\boldsymbol{R}_{\mathrm{couple},i}$ with randomly generated values.

Additionally, disturbances in neural states should have a greater impact earlier in the process before the neural states stabilize (Kopec et al., 2015). Thus, the effect of the coupling disturbance gradually decreases. The formula used is as follows:

$$\boldsymbol{X}_{\mathrm{C\_couple},i}=\boldsymbol{C}_{\mathrm{couple},i}\circ\boldsymbol{R}_{\mathrm{couple},i}\circ\boldsymbol{X}_{\mathrm{couple},i}\cdot r_4\,(1-E)\,, \tag{31}$$

where r_4 is a random number in $[0,1]$, and $E = iteration/Iteration$ is an effect function evaluation number. $Iteration$ denotes the maximum number of iterations, and $iteration$ denotes the present number of iterations. As the iterative process proceeds, $iteration$ gradually increases, whereas the effect of the coupling disturbance gradually decreases.

Eventually, the neural state of each neural population is affected by both the attractors and the neural states of other neural populations. The neural state of the current neural population $\boldsymbol{X}_i$ is updated as follows:

$$\boldsymbol{X}_{\mathrm{new},i}=\lfloor\boldsymbol{X}_i+\boldsymbol{X}_{\mathrm{C\_attract},i}+\boldsymbol{X}_{\mathrm{C\_couple},i}\rfloor\,, \tag{32}$$

where $\lfloor\boldsymbol{X}_i+\boldsymbol{X}_{\mathrm{C\_attract},i}+\boldsymbol{X}_{\mathrm{C\_couple},i}\rfloor$ represents rounding up of elements, because the state of neuron is integer.

Algorithm 3 shows the specific process of the VCPS algorithm for solving the SCSCP problem via three strategies. The inputs consist of the virtual trust matrix $\boldsymbol{M}_{\mathrm{VT}}(t)$ and the physical trust matrix $\boldsymbol{M}_{\mathrm{PT}}(t)$ updated by the trust update algorithm, the physical network G, the SFC request s, the virtual trust weight w_2, along with relevant algorithm parameters (line 1 in Algorithm 3). First, the VCPS algorithm randomly initializes the neural states of n_{p} neural populations (line 3 in Algorithm 3). At each iteration, the dynamics of all neural populations are then calculated, and the top $\lfloor a\cdot n_{\mathrm{p}}\rfloor$ are regarded as attractors (lines 5–6 in Algorithm 3). The neural populations contain the selected VNFs and their placed PNs. Subsequently, the mapped PLs connecting the VNFs are determined using the Dijkstra algorithm, with the cost of unit bandwidth as the link weight, thus enabling the calculation of the dynamics of neural populations. Then each neural state executes the three strategies, but the $\lfloor a\cdot n\rfloor$ attractors are not affected by these three strategies (lines 7–11 in Algorithm 3). The attractor trending strategy is executed, involving assigning an attractor to the neural state and calculating the attractor trending $\boldsymbol{X}_{\mathrm{attract},i}$ (lines 12–13 in Algorithm 3). The coupling disturbance strategy is executed, including calculating the additive coupling $\boldsymbol{X}_{\mathrm{add},i}$, calculating the diffusive coupling $\boldsymbol{X}_{\mathrm{dif},i}$, and obtaining the coupling

Algorithm 3 VCPS algorithm

1: **Input:** physical network G, SFC request s, VNF set F, trust matrices $\boldsymbol{M}_{\mathrm{VT}}(t)$ and $\boldsymbol{M}_{\mathrm{PT}}(t)$, virtual trust weight w_2; parameters $n, a, f_{\mathrm{attract}}, f_{\mathrm{couple}}$, and $Iteration$
2: **Output:** solution
// Initialize
3: Initialize the neural state of n_{p} neural populations P
4: **while** $iteration \le Iteration$ **do**
5: Calculate dynamics of neural populations using Eq. (15)
6: Determine $\lfloor a\cdot n_{\mathrm{p}}\rfloor$ neural states with the optimal dynamics as the attractor set ATT
7: **while** $i \le n_{\mathrm{p}}$ **do**
8: **if** $i \le \lfloor a\cdot n_{\mathrm{p}}\rfloor$ **then**
9: $\boldsymbol{X}_{\mathrm{new},i}=\boldsymbol{X}_i$
10: Continue
11: **end if**
// Determine the attractor set
12: Randomly select $\mathbf{att}_k \in$ ATT as attractor
// Execute attractor trending strategy
13: Calculate attractor trending $\boldsymbol{X}_{\mathrm{attract},i}$ using Eq. (26)
// Execute coupling disturbance strategy
14: Calculate the additive coupling $\boldsymbol{X}_{\mathrm{add},i}$ using Eq. (27)
15: Calculate the diffusive coupling $\boldsymbol{X}_{\mathrm{dif},i}$ using Eq. (28)
16: Calculate coupling disturbance $\boldsymbol{X}_{\mathrm{couple},i}$ using Eq. (29)
// Execute information projection strategy
17: Randomly generate $\boldsymbol{C}_{\mathrm{attract}}, i$ and $\boldsymbol{R}_{\mathrm{attract},i}$
18: Calculate $\boldsymbol{X}_{\mathrm{C\_attract},i}$ using Eq. (30)
19: Randomly generate $\boldsymbol{C}_{\mathrm{couple},i}$ and $\boldsymbol{R}_{\mathrm{couple},i}$
20: Calculate $\boldsymbol{X}_{\mathrm{C\_couple},i}$ using Eq. (31)
21: Calculate the new neural state $\boldsymbol{X}_{\mathrm{new},i}$ using Eq. (32)
// Check and adjust neural state
22: Check whether neuron states meet the state space
23: Adjust the states of neurons to meet the state space
24: $i = i+1$
25: **end while**
26: **end while**
27: Select optimal neural state as Solution
28: **return** Solution

disturbance $\boldsymbol{X}_{\text{couple},i}$ from $\boldsymbol{X}_{\text{add},i}$ and $\boldsymbol{X}_{\text{dif},i}$ (lines 14–16 in Algorithm 3). Information projection onto the above dynamic attractor trending and coupling disturbance is carried out to obtain the newly generated neural state $\boldsymbol{X}_{\text{new},i}$ for the next iteration (lines 17–21 in Algorithm 3). To ensure the feasibility of the neural states, the states of all neurons in $\boldsymbol{X}_{\text{new},i}$ are checked and adjusted. This is accomplished by randomly selecting from the state space for error neurons (lines 22–23 in Algorithm 3). Until the end of the iteration, the best neural state is selected as the optimal solution for the SFC request (line 27 in Algorithm 3).

6 Performance evaluation

6.1 Simulation setup

Obtaining historical cooperation data is challenging. To simplify trust calculation and dynamic trust matrix updates during simulation, we assume that all SFC requests are deployed simultaneously at time t, eliminating the need for real-time updates of the virtual and physical trust matrices. Direct trust values among entities are pre-assigned within [0.5, 1], bypassing the need for historical data. Although the trust matrix is static, this does not compromise evaluation credibility, as the setup still effectively assesses the algorithm's performance.

The physical network adopts the Germany50 topology with 50 nodes and 88 edges (Eramo et al., 2017). Each node has a 48-core CPU, and each edge offers 40 Gbit/s bandwidth with delay uniformly distributed between 2 ms and 4 ms. CPU unit cost ranges in [2, 4], and bandwidth cost in [0.002, 0.004] per Mbit/s (Afrasiabi et al., 2024). Nine VNF types are available, each with [3, 5] functionally equivalent instances, used to compose the SFCs. Direct trust weight is 0.5, and VNF heterogeneity ranges in [0, 1].

We simulate 100 SFC requests, each containing three to seven VNFs. Each VNF requires [1, 4] CPU units, and request bandwidth is drawn from {100, 150, 200, 250, 300} Mbit/s following a Zipf distribution (Kikuchi and Takahashi, 2016). Default parameters include the following: attractor trending and coupling disturbance scaling factor of 0.7, and virtual trust weight of 0.5. Simulations are conducted on an Intel Core i7-7740 @ 3.60 GHz CPU with 64 GB RAM, using PyCharm for implementation.

6.2 Performance analysis

We first analyze the convergence of the proposed algorithm in terms of different scaling factors. The scaling factors of attractor trending and couple disturbance all vary within the range [0, 1] with a step size of 0.1. Four network topologies of varying sizes are selected as the physical network, namely Germany50, Interroute, Colt Telecom, and Cogent.

Fig. 4 presents the curve of Trust2Cost when $f_{\text{attractor}} = 0.7$ and $f_{\text{coupling}} = 0.7$ and the range of Trust2Cost under different scaling factors at each iteration. It can be observed that regardless of the specific combination of scaling factors, the algorithm can achieve convergence at around 1000 iterations. Additionally, all the values of unit-cost trust at each iteration fall within the shaded area, demonstrating that the JTCGO algorithm exhibits strong robustness and stable convergence behavior regardless of the specific parameter choices. The default parameter combination ($f_{\text{attractor}} = 0.7$ and $f_{\text{coupling}} = 0.7$) not only yields competitive performance but also lies within a representative and typical range of the parameter space, so it is set as the default parameter. In summary, the JTCGO algorithm possesses stable convergence ability and strong robustness.

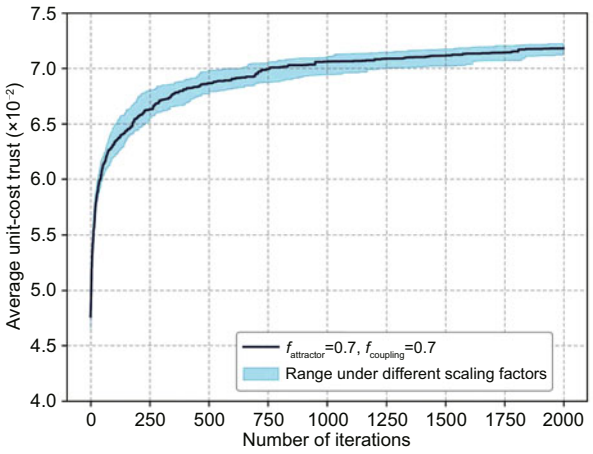

Fig. 4 Convergence of the JTCGO algorithm for different scaling factors

Then, we analyze the impact of network size on the algorithm to prove its scalability. Figs. 5a, 5b, and 5c respectively present the average trust Trust, average cost Cost, and average unit-cost trust Trust2Cost of the 100 SFC requests in four differently-sized networks. As is evident from Fig. 5a, the average trust rises with the growth of

the network size, yet the trend is not pronounced. This is because, as the network size expands, there are more candidate VNF placement schemes, rendering it more probable to obtain solutions with higher physical trust. The cost experiences a significant increase as the network size grows, as depicted in Fig. 5b. With the increase of the network size, more resources, especially bandwidth resources, are consumed for VNF placement and link mapping, which leads to a substantial cost increment. Additionally, Trust2Cost continuously declines with the increase of the network size, and the reasons can be explained by the trends of trust and cost. No matter which network is used, the proposed algorithm can guarantee the high trust of the SFCs under the condition of controllable costs, with the values mostly concentrated in the range of 0.67–0.80. Furthermore, we evaluate the runtime of the algorithm on four networks of different sizes. The proposed algorithm takes approximately 0.095, 0.16, 0.20, and 0.35 s for one iteration on the four networks, respectively. In the Germany50 network, the runtime is as low as 0.095 s, and even in the Cogent network, the runtime remains within 0.35 s. It can be observed that the runtime of the algorithm does not exhibit exponential growth with increasing network size. Even on large-scale networks, the algorithm demonstrates excellent performance in terms of runtime, which supports the applicability of the proposed algorithm in real-world scenarios. This indicates that the proposed algorithm exhibits excellent performance in terms of scalability.

Furthermore, we analyze the impact of virtual trust weight w_2 and physical trust weight $1-w_2$ on the algorithm. Figs. 6a, 6b, and 6c show the average trust Trust, average cost Cost, and average unit-cost trust Trust2Cost of the service under different virtual trust weights, respectively. We can observe from Fig. 6b that regardless of the virtual trust weight, cost remains relatively stable, concentrating within the range of $[12.7, 13.6]$. However, trust is significantly influenced by the virtual trust weight. Within the value range of $[0, 1]$, it shows a trend of being larger at both ends and smaller in the middle, in addition to being larger on the left side

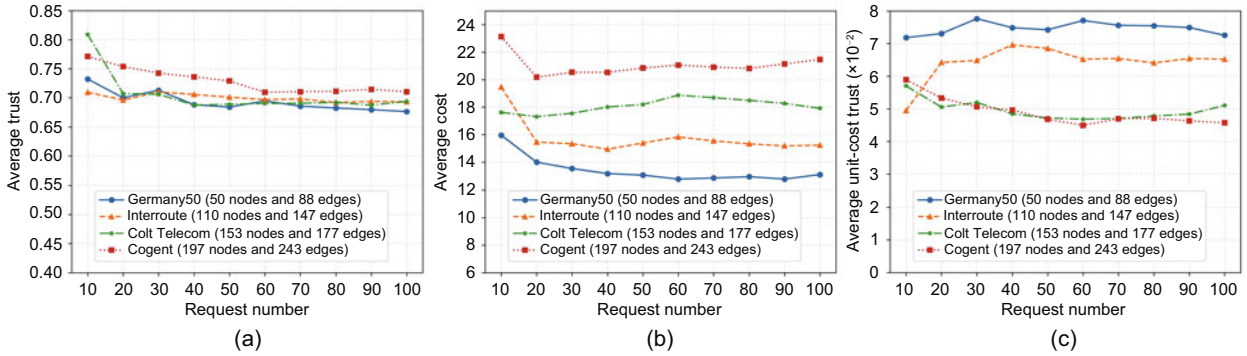

Fig. 5 Performance of the JTCGO algorithm for different networks: (a) average trust; (b) average cost; (c) average unit-cost trust

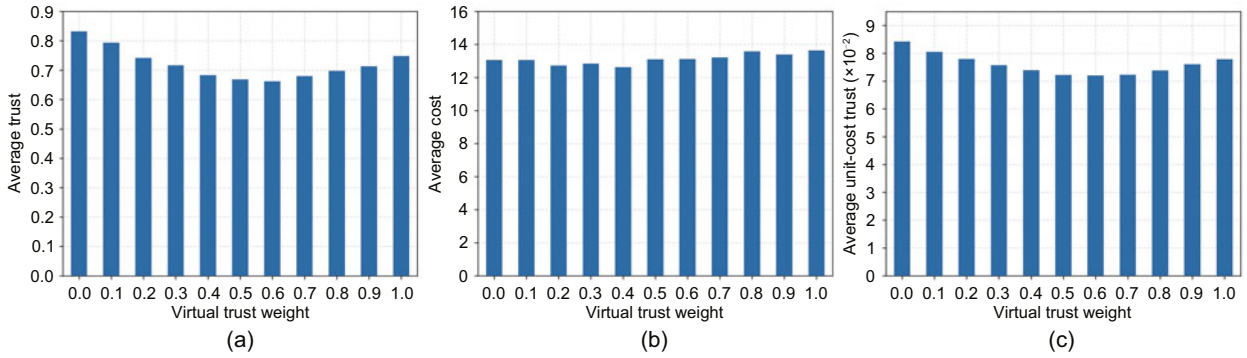

Fig. 6 Performance of the JTCGO algorithm under different weights: (a) average trust; (b) average cost; (c) average unit-cost trust

and smaller on the right side, as shown in Fig. 6a. Under the condition that the cost is approximately the same, Trust2Cost follows the same trend as trust, as shown in Fig. 6c. To explore the reasons behind these trends, we conduct a further analysis of virtual trust and physical trust of the services, as presented in Fig. 7. As the virtual trust weight steadily increases, the proposed algorithm places more emphasis on virtual trust. Consequently, virtual trust gradually rises, while the physical trust behaves in the opposite manner. Additionally, a larger number of candidate PNs result in more schemes for SFC placement, which makes it more likely to obtain a higher physical trust. Overall, physical trust is higher than virtual trust. When virtual trust weight w_2 equals 1, virtual trust of the services reaches its maximum value of 0.74. When w_2 is 0, physical trust attains its peak value of 0.83. This further clarifies the trends depicted in Figs. 6a and 6c. In conclusion, the ability to disrupt the attack chains from both the virtual layer and the physical layer can be adjusted by appropriately tuning the virtual trust weight w_2, making the algorithm adapt to a more complex network security environment.

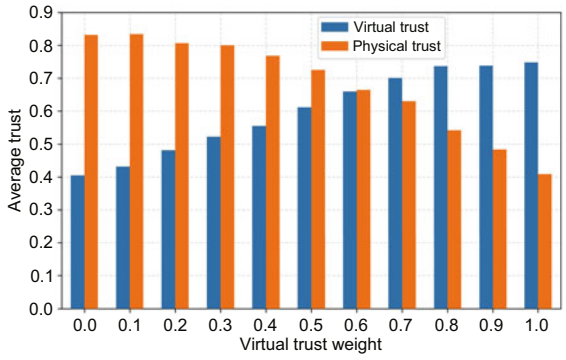

Fig. 7 Average virtual and physical trust under different virtual trust weights

6.3 Performance comparison

To comprehensively evaluate the proposed algorithm, we benchmark it against the existing approaches under various scenarios. To the best of our knowledge, no prior work directly addresses the SCSCP problem; thus, we adopt four relevant algorithms for comparison: BGWO-LT (Shahjalal et al., 2022), HARS-PT (Niu et al., 2022), PB-TASCE (Torkzaban and Baras, 2020), and PSO-C (Gherari et al., 2024). BGWO-LT (Shahjalal et al., 2022) first selects the VNF combination with the maximum virtual trust, and then uses the BGWO algorithm to find the placement that maximizes unit-cost trust. HARS-PT (Niu et al., 2022) maximizes physical trust and uses the HARS method for joint VNF selection and placement. PB-TASCE (Torkzaban and Baras, 2020) identifies the highest virtual-trust VNF combination, generates 100 lowest-cost candidate paths, and selects the one with the maximum physical trust. PSO-C (Gherari et al., 2024) randomly selects one VNF per function type to form a combination, and then applies PSO to minimize total cost during placement. Furthermore, to demonstrate the superiority of the proposed algorithm, we use it to obtain solutions with the maximum trust and minimum cost respectively, denoted as JTCGO-T and JTCGO-C.

Fig. 8a presents the average trust comparison of different algorithms. The trust of JTCGO-T is the highest, with its average trust reaching 0.91. Although BGWO-LT and HARS-PT find solutions to maximize virtual trust and physical trust respectively, their overall trust levels remain lower than that of JTCGO-T. This is because solely pursuing the maximization of either virtual trust or physical trust does not necessarily result in the maximum overall trust, which thereby demonstrates the global optimization ability of the proposed algorithm. JTCGO aims to maximize the unit-cost trust, but its trust value is still approximately on a par with that of HARS-PT and is higher than that of PB-TASCE. Both PSO-C and JTCGO-C sacrifice trust to pursue cost, resulting in the worst cost performance, but the trust of JTCGO-C is still better than that of PSO-C.

Fig. 8b shows the average cost comparison of different algorithms. Both JTCGO-C and PSO-C perform well in terms of cost. Although both aim to minimize the cost, JTCGO-C still outperforms PSO-C in cost, and the average cost reaches the lowest 9.62. Although JTCGO aims to maximize the unit-cost trust, it still outperforms BGWO-LT, HARS-PT, JTCGO-T, and PB-TASCE in terms of cost. PB-TASCE determines the candidate paths with the minimum unit cost. However, its overall cost turns out to be the largest because it needs to consider the service trust when selecting the best candidate path as the solution. Therefore, although the solution is determined from the candidate paths

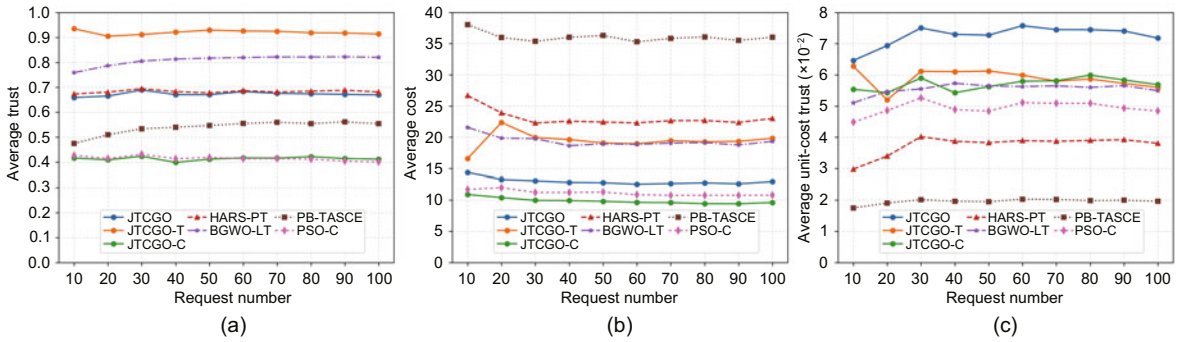

Fig. 8 Performance comparison of different algorithms: (a) average trust; (b) average cost; (c) average unit-cost trust

with the minimum unit cost, PB-TASCE cannot guarantee that VNFs are placed on PNs with the lowest unit cost, resulting in the difficulty of ensuring the service cost. This further illustrates the difficulty in solving the SCSCP problem.

Fig. 8c shows the average unit-cost trust of different algorithms. By combining Figs. 8a and 8b, it can be seen that the JTCGO algorithm exhibits the best performance, with an average unit-cost trust reaching 7.18×10^{-2}. Although PSO-C performs well in terms of cost, it is difficult to achieve the balance between trust and cost. Compared with BGWO-LT and HARS-PT, which also take only trust or cost into consideration, JTCGO-C and JTCGO-T achieve a higher unit-cost trust. PB-TASCE performs the poorest among them.

In summary, JTCGO, JTCGO-T with the objective of trust, and JTCGO-C with the objective of cost demonstrate the best performance in terms of unit-cost trust, trust, and cost, respectively. Based on the above comparison results, it can be concluded that the proposed algorithm JTCGO possesses a stronger ability to solve the SCSCP problem and achieve the trade-off of trust and cost, thereby providing the optimal solution for SFC requests. Consequently, the proposed JTCGO algorithm can ensure service trust at a controllable cost, safeguarding SFCs from network attacks by disrupting the attack chains in a complex security environment.

7 Conclusions

In this paper, we prevent network attacks from occurring by disrupting the attack chains in the virtual layer and physical layers of SFCs. Vulnerability correlation places a specific requirement on the vulnerabilities within the attack chains. Specifically, it demands that the exploitation outcome of an earlier vulnerability serves as a prerequisite for the exploitation of a subsequent vulnerability. The differences in vulnerabilities among network entities lead to the fact that the VNF combination at the SFC composition stage and PN selection at the SFC placement stage directly affect whether the attack chains in SFCs are disrupted. Therefore, we propose the hierarchical trust model to quantify the likelihood of existence of attack chains between network entities, which considers the impact of vulnerability correlation and the vulnerability differences among network entities on the attack chains. Based on the trustworthiness assessment, both VNF combination at the SFC composition stage and PN selection at the SFC placement stage are considered, and the SCSCP problem is formulated as an NP-hard problem, which disrupts the attack chains in SFCs as much as possible at a controllable cost. We propose the JTCGO algorithm including the trust update algorithm and the VCPS algorithm to globally find the SFC deployment solutions for SFC requests. Extensive evaluations and comparisons demonstrate the effectiveness of the proposed algorithm in obtaining higher trust at a controllable cost. Our work prevents the launching of attacks on SFCs at the source and makes up for the inadequacy of the existing works, which can merely act as attack mitigation methods.

Contributors

Deqiang ZHOU and Xinsheng JI designed the research. Deqiang ZHOU, Hang QIU, and Jie YANG processed the data. Deqiang ZHOU drafted the paper. Wei YOU helped organize the paper. Yu ZHAO and Mingyan XU revised and finalized the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Afrasiabi SN, Ebrahimzadeh A, Promwongsa N, et al., 2024. Cost-efficient cluster migration of VNFs for service function chain embedding. *IEEE Trans Netw Serv Manag*, 21(1):979-993. https://doi.org/10.1109/TNSM.2023.3287757

Alomari Z, Zhani MF, Aloqaily M, et al., 2023. On ensuring full yet cost-efficient survivability of service function chains in NFV environments. *J Netw Syst Manag*, 31(3):45. https://doi.org/10.1007/s10922-023-09734-3

Bagheri A, Shameli-Sendi A, 2023. Automating the translation of cloud users' high-level security needs to an optimal placement model in the cloud infrastructure. *IEEE Trans Serv Comput*, 16(6):4580-4590. https://doi.org/10.1109/TSC.2023.3327632

Cao HT, Jindal A, Hu H, et al., 2022. Secure and intelligent service function chain for sustainable services in healthcare cyber physical systems. *IEEE Trans Netw Sci Eng*, 10(5):2674-2684. https://doi.org/10.1109/TNSE.2022.3189546

Cao HT, Yang LX, Garg S, et al., 2024. Softwarized resource allocation of tailored services with zero security trust in 6G networks. *IEEE Wirel Commun*, 31(2):58-65. https://doi.org/10.1109/MWC.001.2300383

Eramo V, Miucci E, Ammar M, et al., 2017. An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures. *IEEE/ACM Trans Netw*, 25(4):2008-2025. https://doi.org/10.1109/TNET.2017.2668470

Gherari M, Dieye M, Elbiaze H, et al., 2024. 3C resource allocation for next-generation applications in an in-network computing-enabled edge-cloud continuum. Proc IEEE Global Communications Conf, p.614-619. https://doi.org/10.1109/GLOBECOM52923.2024.10901375

Hasneen J, Sadique KM, 2022. A survey on 5G architecture and security scopes in SDN and NFV. In: Iyer B, Ghosh D, Balas VE (Eds.). Applied Information Processing Systems. Advances in Intelligent Systems and Computing, Springer, Singapore, p. 447-460. https://doi.org/10.1007/978-981-16-2008-9_43

Herrera JG, Botero JF, 2016. Resource allocation in NFV: a comprehensive survey. *IEEE Trans Netw Serv Manag*, 13(3):518-532. https://doi.org/10.1109/TNSM.2016.2598420

Hong J, Park S, Yoo JH, et al., 2020. A machine learning based SLA-aware VNF anomaly detection method in virtual networks. Proc Int Conf on Information and Communication Technology Convergence, p.1051-1056. https://doi.org/10.1109/ICTC49870.2020.9289547

Hu Y, Guo YA, 2021. Survivable service function chain mapping in NFV-enabled 5G networks. Proc 7th Int Conf on Network Softwarization, p.375-380. https://doi.org/10.1109/NetSoft51509.2021.9492596

Ji JZ, Wu TX, Yang CC, 2024. Neural population dynamics optimization algorithm: a novel brain-inspired meta-heuristic method. *Knowl-Based Syst*, 300:112194. https://doi.org/10.1016/j.knosys.2024.112194

Jorquera Valero JM, Sánchez Sánchez PM, Gil Pérez M, et al., 2023. Cutting-edge assets for trust in 5G and beyond: requirements, state of the art, trends, and challenges. *ACM Comput Surv*, 55(11):1-36. https://doi.org/10.1145/3572717

Kikuchi H, Takahashi K, 2016. Zipf distribution model for quantifying risk of re-identification from trajectory data. *J Inform Process*, 24(5):816-823. https://doi.org/10.2197/ipsjjip.24.816

Kopec CD, Erlich JC, Brunton BW, et al., 2015. Cortical and subcortical contributions to short-term memory for orienting movements. *Neuron*, 88(2):367-377. https://doi.org/10.1016/j.neuron.2015.08.033

Niu M, Han QM, Cheng B, et al., 2022. HARS: a high-available and resource-saving service function chain placement approach in data center networks. *IEEE Trans Netw Serv Manag*, 19(2):829-847. https://doi.org/10.1109/TNSM.2022.3145103

Pattaranantakul M, Vorakulpipat C, Takahashi T, 2023. Service function chaining security survey: addressing security challenges and threats. *Comput Netw*, 221:109484. https://doi.org/10.1016/j.comnet.2022.109484

Peng CZ, Zheng DY, Philip S, et al., 2021. Latency-bounded off-site virtual node protection in NFV. *IEEE Trans Netw Serv Manag*, 18(3):2545-2556. https://doi.org/10.1109/TNSM.2021.3096477

Peretz R, Shenzis S, Hay D, 2020. Moving target defense for virtual network functions. Proc IEEE/IFIP Network Operations and Management Symp, p.1-9. https://doi.org/10.1109/NOMS47738.2020.9110334

Semedo JD, Zandvakili A, Machens CK, et al., 2019. Cortical areas interact through a communication subspace. *Neuron*, 102(1):249-259. https://doi.org/10.1016/j.neuron.2019.01.026

Shahjalal M, Farhana N, Roy P, et al., 2022. A binary gray wolf optimization algorithm for deployment of virtual network functions in 5G hybrid cloud. *Comput Commun*, 193:63-74. https://doi.org/10.1016/j.comcom.2022.06.041

Tang L, Xue CC, Zhao YC, et al., 2024. Anomaly detection of service function chain based on distributed knowledge distillation framework in cloud-edge Industrial Internet of Things scenarios. *IEEE Int Things J*, 11(6):10843-10855. https://doi.org/10.1109/JIOT.2023.3327795

Torkzaban N, Baras JS, 2020. Trust-aware service function chain embedding: a path-based approach. Proc IEEE Conf on Network Function Virtualization and Software Defined Networks, p.31-36. https://doi.org/10.1109/NFV-SDN50289.2020.9289885

Torkzaban N, Papagianni C, Baras JS, 2019. Trust-aware service chain embedding. Proc 6th Int Conf on Software Defined Systems, p.242-247. https://doi.org/10.1109/SDS.2019.8768602

Valente A, Ostojic S, Pillow JW, 2022. Probing the relationship between latent linear dynamical systems and low-rank recurrent neural network models. *Neur Comput*, 34(9):1871-1892. https://doi.org/10.1162/necoa01522

Varadharajan V, Karmakar KK, Tupakula U, et al., 2022. Toward a trust aware network slice-based service provision in virtualized infrastructures. *IEEE Trans Netw Serv Manag*, 19(2):1065-1082. https://doi.org/10.1109/TNSM.2021.3128882

Vyas S, Golub MD, Sussillo D, et al., 2020. Computation through neural population dynamics. *Annu Rev Neurosci*, 43(1):249-275. https://doi.org/10.1146/annurev-neuro-092619-094115

Wang M, Cheng B, Wang SG, et al., 2021. Availability-and traffic-aware placement of parallelized SFC in data center networks. *IEEE Trans Netw Serv Manag*, 18(1):182-194. https://doi.org/10.1109/TNSM.2021.3051903

Wang WL, Liang CC, Chen QB, et al., 2022. Distributed online anomaly detection for virtualized network slicing environment. *IEEE Trans Veh Technol*, 71(11):12235-12249. https://doi.org/10.1109/TVT.2022.3193074

Wang WL, Liang CC, Tang L, et al., 2023. Federated multi-discriminator BiWGan-GP based collaborative anomaly detection for virtualized network slicing. *IEEE Trans Mob Comput*, 22(11):6445-6459. https://doi.org/10.1109/TMC.2022.3200059

Wang WL, Zhou HC, Li M, et al., 2024. An autonomous deployment mechanism for AI security services. *IEEE Access*, 12:4048-4062. https://doi.org/10.1109/ACCESS.2023.3346187

Yu XH, Jiang JH, Shuai CY, 2013. Approach to attack path generation based on vulnerability correlation. *IEEE Conf Anthol*. https://doi.org/10.1109/ANTHOLOGY.2013.6784925

Zhang PY, Wang C, Jiang CX, et al., 2021. Security-aware virtual network embedding algorithm based on reinforcement learning. *IEEE Trans Netw Sci Eng*, 8(2):1095-1105. https://doi.org/10.1109/TNSE.2020.2995863

Zhang QQ, Tang HB, You W, et al., 2021. Network function heterogeneous redundancy deployment method based on immune algorithm. *Chin J Netw Inform Secur*, 7(1):46-56 (in Chinese). https://doi.org/10.11959/j.issn.2096-109x.2021005

Zhang T, Xu CQ, Zhang BC, et al., 2023. Towards attack-resistant service function chain migration: a model-based adaptive proximal policy optimization approach. *IEEE Trans Depend Secur Comput*, 20(6):4913-4927. https://doi.org/10.1109/TDSC.2023.3237604

Zhang Y, Jiang CX, Zhang PY, 2023. Security-aware resource allocation scheme based on DRI in cloud-edge-terminal cooperative vehicular network. *IEEE Int Things J*, 11(1):95-104. https://doi.org/10.1109/JIOT.2023.3293497

Zheng DY, Liu XR, Tang WY, et al., 2023. Cost optimization in security-aware service function chain deployment with diverse vendors. Proc IEEE Global Communications Conf, p.2093-2098.

Zheng DY, Xing HL, Feng L, et al., 2024. Provably efficient security-aware service function tree composing and embedding in multi-vendor networks. *Comput Netw*, 254:110843. https://doi.org/10.1016/j.comnet.2024.110843

Zhou DQ, Ji XS, You W, et al., 2024. DDQN-SFCAG: a service function chain recovery method against network attacks in 6G networks. *Comput Netw*, 254:110748. https://doi.org/10.1016/j.comnet.2024.110748