drones

Article

Research on Service Function Chain Embedding and Migration Algorithm for UAV IoT

Xi Wang, Shuo Shi * and Chenyu Wu

School of Electronic and Information Engineering, Harbin Institute of Technology, Harbin 150001, China; wangxi_chn@foxmail.com (X.W.); wuchenyu@hit.edu.cn (C.W.)
* Correspondence: crcss@hit.edu.cn

Abstract: This paper addresses the challenge of managing service function chaining (SFC) in an unmanned aerial vehicle (UAV) IoT, a dynamic network that integrates UAVs and IoT devices for various scenarios. To enhance the service quality and user experience of the UAV IoT, network functions must be flexibly configured and adjusted based on varying service demands and network situations. This paper presents a model for calculating benefits and an agile algorithm for embedding and migrating SFC based on particle swarm optimization (PSO). The model takes into account multiple factors such as SFC quality, resource utilization, and migration cost. It aims to maximize the SFC benefit and minimize the migration times. The algorithm leverages PSO's global search and fast convergence to identify the optimal or near-optimal SFC placement and update it when the network state changes. Simulation experiments demonstrate that the proposed method improves network resource efficiency and outperforms existing methods. This paper presents a new idea and method for managing SFC in UAV IoT.

Keywords: UAV IoT; NFV; service function chaining; network resource efficiency

check for updates

Citation: Wang, X.; Shi, S.; Wu, C. Research on Service Function Chain Embedding and Migration Algorithm for UAV IoT. *Drones* **2024**, *8*, 117. https://doi.org/10.3390/drones8040117

Academic Editors: Michel Kadoch and Bo Rong

Received: 14 February 2024
Revised: 19 March 2024
Accepted: 21 March 2024
Published: 22 March 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

1.1. Background and Research Motivation

With the rapid development of wireless communication technology, unmanned aerial vehicles (UAVs), as flexible, efficient, and low-cost air platforms, have become widely utilized in civil, military, commercial, and other fields [1]. UAVs can be equipped with a variety of IoT devices, such as sensors, cameras, and GPS modules, forming an airborne IoT system that provides a range of services to users on the ground, including monitoring, communication, and sensing [2]. However, the dynamic, heterogeneous, and distributed nature of UAV networks brings significant challenges for network management and optimization. In order to enhance the service quality and user experience of UAV networks, it is essential to flexibly configure and adjust the network functions to meet various business requirements and network conditions [3].

Network virtualization is a technology that implements multiple virtual networks on the same physical network to improve the utilization, flexibility, and innovation of network resources [4]. Network virtualization enables the software and programmability of network functions, reduces network costs and complexity, and improves the delivery speed and quality of network services. By employing network virtualization technology, network functions can be virtualized into VNFs (virtualized network functions), which can be deployed at various layers of the UAV network to achieve quick configuration, adjustment, and migration of network functions. This enhances the performance, reliability, and security of the UAV network [5].

Based on VNFs and software-defined networking (SDN) technologies, service function chaining (SFC) connects network functions in a specific sequence to achieve flexible orchestration and scheduling of network functions, adapting to different service scenarios and network states, as shown in Figure 1. Because of its network characteristics, which combine UAVs and IoT devices, it can support a variety of complex business scenarios, such as emergency rescue, intelligent transportation, and smart agriculture [5]. These service scenarios have diverse network requirements, such as high bandwidth, low latency, and high reliability. In order to meet these requirements, network functions need to be carefully controlled and optimized to achieve on-demand combination and customization of network functions [6]. Therefore, the SFC has significant application value and research significance in UAV IoT.

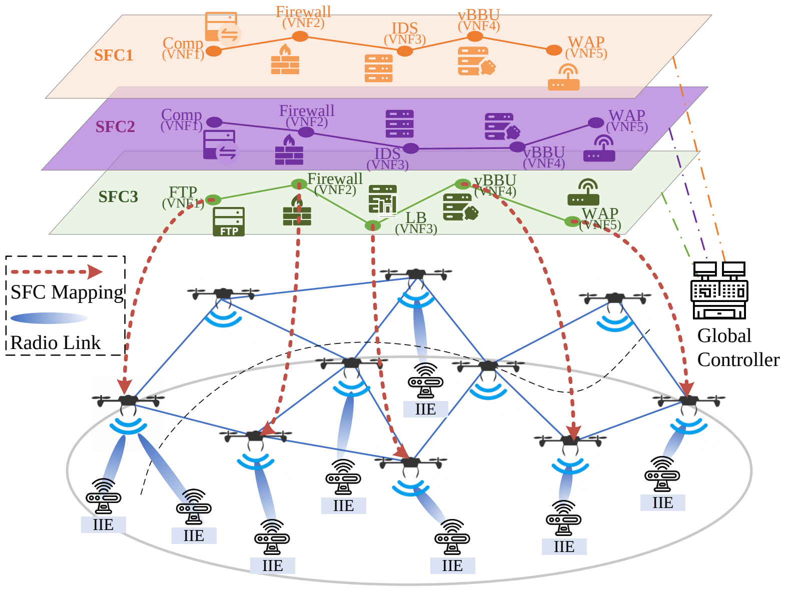

Figure 1. Deployment of VNFs on the UAV IoT to enable network slicing and support the SFC.

1.2. Network Architecture and Challenges

The network architecture we used for the UAV IoT leverages SFC based on VNF and SDN technologies, as shown in Figure 1. This design enables a dynamic and flexible orchestration of network functions to cater to varying service scenarios and network states.

System Overview: A fundamental component of the architecture is a set of UAVs, each equipped with a common computing platform capable of hosting multiple virtual machines (VMS). These virtual machines provide a deployment environment for SFC services requested by terrestrial Industrial IoT equipment (IIE) to support the transmission of service data.

Network Resources: We consider the four most typical node resources provided by the UAV, namely, the computing resources (CPU), represented by the number of CPU cores; the memory resources (RAM), represented by the size of the cache space; the storage resources (DISK), represented by the size of the disk space; and the energy consumption resources (ENG), represented by the battery capacity on the UAV. Without loss of generality, we consider the most typical transmission bandwidth (BAND), that is, the maximum rate of data transmission between two UAVs.

Global Controller: Within the network's deployment domain, the global controller acts as a command and communications hub, coordinating the entire UAV network. It is responsible for policy implementation of SFC embedding and migration across UAVs. The controller can be deployed for ground stations or spaceborne platforms.

SFC Embedding Workflow: When the IIE sends an SFC setup request to the UAV network, the workflow starts. After receiving the request, the UAV relays the request to the VNF manager located in the global controller. According to the resource quantity and quality of service (QoS) requirements in the request, the VNF manager uses the embedded algorithm to divide node and link resources on the computing platform of each UAV and establish SFC in the UAV network. Then, SDN technology is used to configure network routing and manage the forwarding of service data.

Real-Time Monitoring and Adaptation: The global controller continuously monitors the performance of SFC on the network in real time. The SFC migration algorithm is triggered when the network topology changes and the link fails or the QoS fails to meet the SFC requirements. This process includes invalidating the existing VNFs, removing the existing VNFs, establishing a new VNFS, and then rerouting to complete the establishment of the new SFC.

The architecture is designed to provide a robust, adaptable, and efficient communication network for UAV IoT systems, capable of supporting complex IIE data transfers with high reliability and flexibility. The research in this paper is based on two observations in this network architecture:

(1) SFC has life cycle characteristics and requires maintenance during operation. Its life cycle includes creation, embedding, operation, migration, and destruction. Due to the dynamic characteristics of the network, such as UAV movement, network topology changes, and resource fluctuations, system performance may deteriorate or services may even be interrupted. Therefore, real-time monitoring and maintenance of the SFC are essential to guarantee the availability and reliability of the SFC.

(2) Due to the particularity of UAV energy consumption, load balancing should be considered in the embedding algorithm. If the SFC embedding algorithm does not consider the energy consumption resources of UAVs, it may lead to the premature depletion of UAVs' energy or an imbalance of energy consumption among UAVs, thus affecting the stability and sustainability of UAVs' dynamic network.

Based on the above observations, the challenge of this research is to design an SFC embedding algorithm deployed on a global controller. The algorithm can dynamically select the appropriate UAV as the SFC execution node according to the energy consumption resources and network status of the UAV, so as to achieve efficient SFC embedding and balance the energy consumption of the UAV. At the same time, in order to ensure the service continuity during the SFC life cycle, the algorithm can dynamically migrate SFC according to the real-time performance of SFC and the energy distribution of the UAVs.

1.3. Contributions

The main contributions of this paper are as follows:

- Aiming to address the resource perception problem of SFC in UAV dynamic networks, a revenue calculation model based on SFC operation process monitoring is proposed. This model comprehensively considers the service quality, resource utilization, and migration cost of SFC.
- An agile algorithm for embedding and migrating SFC based on particle swarm optimization (PSO) is proposed. It can quickly find the optimal or nearly optimal embedding scheme to meet the requirements of SFC and facilitate the agile migration of SFC when the network state changes.
- A simulator and simulation environment based on MANO architecture has been developed. It is capable of simulating dynamic network topology, SFC operation processes, embedding, and migration processes.
- The experimental results demonstrate that the proposed method outperforms the comparison algorithm in terms of SFC income, migration quality, and resource utilization, thus confirming the effectiveness and superiority of the proposed method.

2. Related Work

In this paper, from the perspective of network operators, we focus on the SFC embedding and migration problem of UAV dynamic network topology under the scenario of a large number of SFC service requests. This section will introduce some representative and high-quality related studies from recent years.

2.1. SFC Embedding

The software implementation of VNFs provides the flexibility for service providers to deploy SFC in the existing networks. Therefore, a critical issue to be solved is how to determine the deployment location of each VNF in the SFC, which is called the SFC embedding problem.

Beck et al. [7] took the lead in bringing this problem to researchers' attention and proposed a universal virtual network framework that embedded multiple virtual networks onto the base network. On this basis, a heuristic algorithm is used to coordinate the embedding of VNF chains with the goal of minimizing bandwidth utilization [8]. However, the framework is oriented to the large-capacity server of the computing center and cannot be directly applied to the UAV computing platform. With the goal of minimizing the total deployment cost, Tomassilli et al. [9] designed an optimal algorithm based on tree topology to optimally place VNFs when meeting all SFC requirements of the stream. Sallam et al. [10] expressed the maximum flow problem constrained by SFC as the fractional multi-commodity flow problem and then solved it. Jin et al. [11] expressed the VNF chain deployment problem as mixed-integer linear programming (MILP) to minimize total resource consumption in the context of computing service deployment on the edge service area. However, due to their dependence on prior knowledge and high computational complexity, these offline methods are difficult to apply to online placement scenarios.

In recent years, some researchers have turned their attention to hybrid SFC embedding problems, such as bidirectional SFC based on real-world business needs. Zheng et al. [12] define a new problem for minimum delay hybrid SFC embedding (ML-HSFCE) and propose an optimal hybrid SFC embedding algorithm (Opt-HSFCE). They extended this work in [13], proposing the first 2-approximation algorithm to jointly optimize the processes of h-SFC construction and embedding, which is called Eulerian-circuit-based hybrid SFP optimization (EC-HSFP). Dimolitsas et al. [14] proposed a distributed VNF embedding method (DVNE) to embed two SFC round-trips with the goal of minimizing round-trip latency. The problem of hybrid SFC embedding focuses on the service delay after the completion of embedding, and the service delay in these studies is mainly described by the hop number in the middle of the node, which does not accord with the fact that a large amount of delay also exists in the virtual resource scheduling on each computing platform.

2.2. SFC Migration

The embedding of SFC in ground infrastructure has been widely studied, but the research progress on the embedding of SFC in a platform with high mobility, such as a UAV network, is relatively slow. Although Vidal et al. [15] has built a UAV network test platform supporting SFC to verify its effectiveness, its highly dynamic and changing topology poses challenges to the real-time migration effect of SFC services.

Carpio et al. [16] investigated the problem of optimal placement of hybrid SFC from an Internet service provider (ISP) perspective and proposed a two-stage optimization process to analyze the performance impact of replication and migration of VNF. They extended the work using a traffic forecasting approach [17] to reduce migration costs. Rui et al. [18] modeled the SFC as a Petri net model and obtained reliability evaluation results related to execution time. Then, a VNF migration strategy with reliability as the optimization objective and cost as a consideration was designed. While the above study noted the importance of SFC migration on cloud computing platforms, it focused more on migrating SFC to accommodate the dynamic changes in services traffic, which is different from the dynamic nature of deploying SFC on UAV networks.

Qin et al. [3] described the application scenario of SFC for vehicle-assisted or UAV-assisted edge computing, studied the SFC migration problem in dynamic networks with long-term cost budget constraints, and used the Lyapunov optimization method to solve it. In the same scenario, Bai et al. [19] used a quantitative modeling method based on semi-Markov process to study the potential impact of VNF migration time and network elasticity in SFC. In order to maximize the acceptance rate of service function chain requests (SFCRs) and reduce VNF migration and energy consumption as much as possible, Hu et al. [20] summarized the relevant factors such as node status, link status, energy consumption, migration node, and mapping success. Then, the VNF migration algorithm was designed using the actor-critic model, graph convolutional network, and LSTM network.

The optimization goal of these studies is to maximize the success rate of SFC embedding and improve the system benefits. However, considering that there is a life cycle of SFC, the revenue calculation of SFC cannot be determined only by whether it completes the embedding, and the supervision in its operation process is equally important, which is missing in the above research.

3. System Model

In this section, we describe the system model in five aspects, including the substrate network, the service function chain, the SFC embedding, the SFC latency, and the SFC revenue. Then, we formulate the problem as an integer programming model. The symbols used in this paper are summarized in Table 1.

Table 1. Notation and definitions.

Notation	Definition
N_S	Set of substrate network nodes
n_S^i	Each physical node in substrate network
$L_S(t)$	Set of substrate network links at time t
$l_S^{i,j}(t)$	Each physical link between node n_S^i and node n_S^j at time t
$C_S(n_S)$	Maximum resource capacity of node n_S
$C'_S(n_S,t)$	Remaining resource of node n_S at time t
$C_S(l_S)$	Maximum resource capacity of link l_S
$C'_S(l_S,t)$	Remaining resource of link l_S at time t
$\mathbf{W}_S(n_S)$	"Wait-Time Matrix" for servers in physical node n_S [1]
$\mathcal{F}$	Set of SFCs to be processed
N_f	Set of virtual network nodes (VNFs) of SFC $f \in \mathcal{F}$
n_f^i	Each virtual node in SFC $f \in \mathcal{F}$
L_f	Set of virtual network links of SFC $f \in \mathcal{F}$
$l_f^{i,j}$	Each virtual link between VNF n_f^i and VNF n_f^j
$C_f(n_f)$	Resource need to be allocated of VNF n_f
$C_f(l_f)$	Resource need to be allocated of virtual link l_f
t_f^a, t_f^d, t_f^e	Arrival time, life cycle, and end time of SFC $f \in \mathcal{F}$
LA_f	Transmission latency requirement of SFC $f \in \mathcal{F}$
$\mathbf{X}_f(t)$	Set of mapping indication between VNF in SFCf and physical node at time t
$x_f^{n_f^i, n_S^j}(t)$	Whether VNF n_f^i is deployed on physical node n_S^j at time t
$\mathbf{Y}_f(t)$	Set of mapping indication between virtual link in SFCf and physical link at time t
$y_f^{l_f^{i,j}, l_S^{i',j'}}(t)$	Whether virtual link $l_f^{i,j}$ is embedded in physical link $l_S^{i',j'}$ at time t
$\psi_f(t)$, $\psi_f^0(t)$, $\psi_f^1(t)$, $\psi_f^2(t)$	Total latency, operation latency, remap and reroute latency of SFC $f \in \mathcal{F}$ at time t
R^f	System revenue of SFC $f \in \mathcal{F}$

[1] Boutin et al. [13] defines the latency matrix for servers.

3.1. Substrate Network Model

The base network is represented as an infrastructure by an undirected graph $G_S = (N_S, L_S)$, with network nodes denoted as $N_S = \{n_S^1, \dots, n_S^i, \dots, n_S^{|N_S|}\}$, and $n_S \in N_S$ for each physical node. Due to topological dynamics, we define L_S as an adjacency matrix, unlike the typical definition of links in graphs, and each element in the matrix is determined by a function of time t. Thus, there is a set of physical links $L_S(t) = \{l_S^{1,1}(t), \ldots, l_S^{i,j}(t), \ldots, l_S^{|N_S|,|N_S|}(t)\}$, and for each physical link $l_S \in L_S$, $l_S^{i,j}$ represents the link between node n_S^i and node n_S^j, where:

$$l_S^{i,j}(t) = \begin{cases} 1, \text{ } if \text{ } physical \text{ } link \text{ } between \text{ } n_S^i \text{ } and \text{ } n_S^j \text{ } exists \\ 0, \text{ } otherwise \end{cases} \tag{1}$$

The physical node serves as a server to provide resources for the deployment of SFC, utilizing the function $C_S(*)$ to describe the maximum resource capacity, and the function $C_S{}'(*, t)$ to describe the remaining resources at time t. In this paper, we consider four types of physical node resources: CPU, RAM, DISK, and ENG. Therefore, for node $n_S \in N_S$, its resource capacity is $C_S(n_S) = \{C_S^{CPU}(n_S), C_S^{RAM}(n_S), C_S^{DISK}(n_S), C_S^{ENG}(n_S)\}$, and the remaining resource at time t is $C_S'(n_S, t) = \{C_S^{CPU'}(n_S, t), C_S^{RAM'}(n_S, t), C_S^{DISK'}(n_S, t), C_S^{ENG'}(n_S, t)\}$. Different from other researchers' definitions of network node resources, this paper includes energy consumption resources. It considers UAVs as deployment platforms for physical nodes, with their batteries providing limited power. This is distinct from CPU, RAM, DISK, and other occupation-release resources, as energy consumption is one-time. The resource for a physical link is the transmission bandwidth (BAND). For link $l_S \in L_S$, its maximum resource capacity is $C_S(l_S) = \{C_S^{BAND}(l_S)\}$. As with node resources, SFC in the network will continue to occupy part of the transmission bandwidth between UAVs, so we need to pay attention to the remaining situation of link resources in real time, so as to facilitate the embedding of the new SFC and realize the full utilization of link resources. The remaining resource at time t is $C_S'(l_S, t) = \{C_S^{BAND'}(l_S, t)\}$. The deployment of VNF on physical nodes requires latency for creation and task queues.

We refer to the "Wait-Time Matrix" proposed by Boutin et al. [21] to describe the time that must be waited to request resources from the computing platform on the UAV. We consider that the DISK resources of a node are determined by the total size of the embedded functional software, which is pre-installed on the platform, while the different SFC only activates the software on demand, and the activation time is very short. What really affects the VNF operation time is the number of CPU cores applied for the SFC and the cache of service data. Therefore, the time waiting matrix is determined by the index of CPU resources and RAM resources, a consideration consistent with Boutin et al. [21] 's cluster design for the Microsoft cloud computing platform. The difference is that the table is designed as the index of the remaining resources of the node. For the sake of generality, we set the index granularity to 1 resource unit. The more remaining resources, the shorter the waiting time, and the tighter the remaining resources, the longer the scheduling time of the resources required to complete the task. We define a latency matrix for servers in each physical node as $\mathbf{W}_S(n_S)$. The matrix is indexed by the current CPU resources and the remaining amount of RAM resources of the node, that is, $\mathbf{W}_S(n_S) = table(C_S^{CPU'}(n_S), C_S^{RAM'}(n_S))$. The value indexed in the table represents the latency required for VNF preparation on the server when requesting resources.

3.2. Service Function Chain Model

We define the set of SFCs to be processed as $\mathcal{F}$. For an SFC $f \in \mathcal{F}$, we utilize the directed graph $G_f = (N_f, L_f)$ to represent the virtual network structure of the service function chain. In this structure, the VNF virtual node is denoted as $N_f = \{n_f^1, \ldots, n_f^i, \ldots, n_f^{|N_f|}\}$, and for each virtual node, there exists $n_f \in N_f$. To align with the physical links in the base network, virtual links are defined using the adjacency matrix L_f. In other words, $L_f = \{l_f^{1,1}, \ldots, l_f^{i,j}, \ldots, l_f{}^{|N_f|,|N_f|}\}$, where each virtual link $l_f \in L_f$, $l_f^{i,j}$ denotes the link between virtual node n_f^i and virtual node n_f^j. Of which:

$$l_f^{i,j} = \begin{cases} 1, \text{ } if \text{ } virtual \text{ } link \text{ } between \text{ } n_f^i \text{ } and \text{ } n_f^j \text{ } exists \\ 0, \text{ } otherwise \end{cases} \tag{2}$$

The advantage of using an adjacency matrix to define a network function chain is that it provides convenience for extending subsequent service functions. Schneider et al. [22] has proposed that the virtual network embedding with complex functions may not always exhibit a perfect chain structure. In some cases, bidirectional networks containing loops may be necessary to support complex applications, such as optimizing network video transmission and inserting advertisements, thereby achieving the network slicing effect.

SFC construction involves requesting resources from the base network to sequentially build VNF virtual nodes and connect them through virtual links. We use the function $C_f(*)$ to describe the amount of resources that SFC $f \in \mathcal{F}$ applies to the base network. The resources utilized are the four types available in the substrate network. For a virtual node $n_f \in N_f$, the allocated resources are denoted as $C_f(n_f) = \{C_f^{CPU}(n_f), C_f^{RAM}(n_f), C_f^{DISK}(n_f)$, and $C_f^{ENG}(n_f)\}$. The purpose of the virtual link is to request transmission bandwidth from the base link. Therefore, for the virtual link $l_f \in L_f$, the requested resource occupation applied for is $C_f(l_f) = \{C_f^{BAND}(l_f)\}$. Li et al. [23] point out that the SFC for achieving complete wireless access functionality for the industrial Internet of things should include a wireless access node WAP_f and industrial Internet of things equipment IIE_f, as shown in Figure 1. In this paper, we assume that all UAV nodes have wireless access functions to enable random access of Internet of things equipment. The functions and configurations of these two nodes do not vary across different SFCs, which is not the focus of this paper. Therefore, they will not be discussed further.

When a user requests resources from the network, the network controller must specify the duration for which the resources will be occupied. This allows the network controller to determine when to release the occupied resources and reallocate them to other users, thereby improving the efficiency of network resource utilization. In order to implement this function, the arrival time of SFC is defined as t_f^a, and the service duration, which is the life cycle of SFC, is defined as t_f^d. Therefore, the end time of service can be calculated as $t_f^e = t_f^a - t_f^d$. This paper focuses on the transmission latency LA_f in the quality of service (QoS) requirements that users submit to the network. It should be noted that in SFC resource allocation, the energy consumption resource $C_f^{ENG}(n_f)$ of virtual node n_f is interrelated with other resources, and its magnitude should not be specified from the user's perspective. Su et al. [24] highlight that in the actual operation of VNFs, CPU energy consumption accounts for a larger share compared to RAM and DISK. We used Su et al.'s assumption [24] (i.e., the mapping relationship between energy consumption resources, CPU, and running time in the resources applied by virtual node n_f is $C_f^{ENG}(n_f) = C_f^{CPU}(n_f) \cdot (t_f^e - t)$). When SFC arrives at $t = t_f^a$, the controller anticipates the future energy consumption of the virtual node based on this, and then selects the appropriate deployment location.

3.3. SFC Embedding Model

For SFC $f \in \mathcal{F}$, at time t, the mapping indication of a virtual node to a physical node is defined as $\mathbf{X}_f(t) = [x_f^{n_f^1, n_S^1}(t), \ldots, x_f^{n_f^i, n_S^j}(t), \ldots, x_f^{|N_f|, |N_S|}(t)]$, where $x_f^{n_f^i, n_S^j}(t) = 1$ represents the mapping of virtual node n_f^i to physical node n_S^j. The mapping indication that defines the virtual link to the physical link is represented as $\mathbf{Y}_f(t) = [y_f^{l_f^{1,1}, l_S^{1,1}}(t), \ldots, y_f^{l_f^{i,j}, l_S^{i',j'}}(t), \ldots, y_f^{l_f^{|N_f|,|N_f|}, l_S^{|N_S|,|N_S|}}(t)]$, where $y_f^{l_f^{i,j}, l_S^{i',j'}}(t) = 1$ represents the mapping of virtual link $l_f^{i,j}$ to physical link $l_S^{i',j'}$.

Since the resources provided by physical nodes are limited, the total resources occupied by VNFs running on them must not exceed the upper limit of available resources. Therefore, the mapping of virtual nodes needs to satisfy the following constraint relations:

$$\sum_{n_f \in N_f} x_f^{n_f,n_S} \cdot C_f(n_f) \leq C'_S(n_S), \forall n_S \in N_S \tag{3}$$

Similarly, because the transmission bandwidth of the physical links between UAVs is limited, the requested bandwidth of the virtual link of SFCf mapped to a physical link cannot be greater than the remaining bandwidth. Therefore, the mapping of virtual links must meet the following constraint:

$$\sum_{l_f \in L_f} y_f^{l_f,l_S} \cdot C_f(l_f) \leq C'_S(l_S), \forall l_S \in L_S \tag{4}$$

We do not consider the scenario where a VNF virtual node in SFCf can be deployed and operated concurrently on multiple physical facilities, that is, the virtual node cannot be mapped to multiple physical nodes. Therefore, there is the following constraint:

$$\sum_{n_S \in N_S} x_f^{n_f,n_S} \leq 1, \forall n_f \in N_f \tag{5}$$

Considering the dynamic nature of network topology, the prerequisite for successful virtual link mapping is the existence of physical links in the base network. Therefore, there is the following constraint:

$$y_f^{l_f,l_S} \leq l_S, \forall l_f \in L_f, l_S \in L_S \tag{6}$$

At the same time, we refer to wang et al. [25], using a flow model approach to ensure that the virtual links mapped from SFC f to the base network traverse the VNF virtual nodes in a specified order. We represent the set of incoming and outgoing links of physical node $n_S \in N_S$ as $I(n_S)$ and $O(n_S)$, respectively. Additionally, $n_f(l_f^d)$ and $n_f(l_f^s)$ represent the destination and source VNF of virtual link $l_f \in L_f$, respectively. According to the flow model, all the outgoing traffic from nodes should be equal to the incoming traffic, and this constraint exists only for the virtual nodes and links to which SFC is mapped. Therefore, the constraint can be expressed as:

$$\sum_{l'_S \in I(n_S)} y_{l'_S}^{l_f} - \sum_{l'_S \in O(n_S)} y_{l'_S}^{l_f} = x_{n_S}^{n_f(l_f^d)} - x_{n_S}^{n_f(l_f^s)}, \forall l_f \in L_f, l'_S \in L_S, n_S \in N_S \tag{7}$$

3.4. SFC Latency Model

3.4.1. Operation Latency

We analyzed the transmission delay of an SFC and discovered that the propagation delay and transmission delay are significantly influenced by the size of the data packet. This size is closely related to the SFC's service type and is also affected by the client and server's sending and receiving mechanisms, which are designed by the user and not taken into account by the operator. As an infrastructure leasing service, the primary concern should be the scheduling latency of running VNFs on each physical node. We provided the model for server latency in physical nodes with the "Wait-Time Matrix" at the end of Section 3.1. Considering that multiple VNFs in the SFC are running in parallel to process the service data flow at time t, the delay bottleneck may occur in one of the VNFs. This can be expressed as:

$$\psi_f^0(t) = \max_{n_f \in N_f} \sum_{n_S \in N_S} x_f^{n_f,n_S}(t) \cdot \mathbf{W}_S(n_S) \tag{8}$$

Mapping can only be completed if the SFC's runtime latency meets its QoS requirements. Therefore, there are the following constraints:

$$\psi_f^0(t) \leq LA_f \tag{9}$$

3.4.2. Remap Latency

When a virtual node mapping needs to be migrated from one physical node to another, the delay primarily arises from requesting resources from the server and retransmitting cached data. We use the indicator vector $\mathbf{m}_{n_f}(t)$ to determine the migration relation of the mapping of virtual node n_f at time t. The calculation method is $\mathbf{m}_{n_f}(t) = [x_f^{n_f,N_S}(t-1)] \oplus [x_f^{n_f,N_S}(t)]$, where $x_f^{n_f,N_S}(t)$ represents the vector composed of the mapping relation of virtual node n_f and all physical nodes N_S at time t, and the symbol $\oplus$ represents the exclusive OR operation between two vectors. The resulting migration delay can be calculated as follows:

$$\psi_f^1(t) = \sum_{n_f \in N_f} \frac{||\mathbf{m}_{n_f}(t)||_0}{2} \cdot \frac{C_f^{RAM}(n_f)}{C_f^{BAND}(l_f^{n'_f,n_f})} \tag{10}$$

where $||\mathbf{m}_{n_f}(t)||_0$ represents the 0-norm of $\mathbf{m}_{n_f}(t)$, which is the count of non-zero elements in the vector, and determines the number of physical nodes involved in the migration process. $C_f^{RAM}(n_f)$ represents the amount of buffered data that needs to be retransmitted to recover the state of virtual node n_f, and $C_f^{BAND}(l_f^{n'_f,n_f})$ represents the bandwidth requested by virtual link $l_f^{n'_f,n_f}$ with virtual node n_f as the destination.

3.4.3. Reroute Latency

We believe that embedding SFC into the virtual network relies on the OpenFlow table delivery mechanism of SDN to establish virtual links. Therefore, it is necessary to sequentially modify the routing tables of physical nodes to facilitate the forwarding of service traffic in SFC. The process of distributing the flow table contributes to the rerouting delay. We utilize the indicator vector $\mathbf{m}_{l_f}(t)$ to determine the migration relation of the mapping of the virtual link l_f at time t. The calculation method is $\mathbf{m}_{l_f}(t) = [y_f^{l_f,L_S}(t-1)] \oplus [y_f^{l_f,L_S}(t)]$, where $y_f^{l_f,L_S}(t)$ represents the vector composed of the mapping relation of the virtual link l_f and all physical links L_S at time t, and the symbol $\oplus$ represents the exclusive OR operation between the two vectors. The resulting migration delay can be calculated as follows:

$$\psi_f^2(t) = \sum_{l_f \in L_f} ||\mathbf{m}_{l_f}(t)||_0 \cdot\ t_0 \tag{11}$$

The expression $||\mathbf{m}_{l_f}(t)||_0$ represents the 0-norm of $\mathbf{m}_{l_f}(t)$, which is the count of non-zero elements in the vector. It is used to calculate the number of physical links involved in the migration process. t_0 represents the latency of modifying the routing table needed to modify a mapped physical link.

In summary, the total delay required for SFC service transmission is:

$$\psi_f(t) = \psi_f^0(t) + \psi_f^1(t) + \psi_f^2(t) \tag{12}$$

3.5. SFC Revenue Model

In contrast to other research studies, this paper utilizes the revenue per unit time method. This method is based on observing the dynamic service mode of SFC. SFC adjusts the mapping scheme in real time based on the network topology during operation, leading to temporary network fluctuations and a potential inability to maintain the service according to the original QoS requirements. Therefore, it is not appropriate to base revenue solely on the success of the embedding. In this paper, we employ a lenient decision method to calculate the revenue during the migration process. This means that there may be QoS nonconformity during the migration process, but it is essential to ensure that the QoS requirements can be satisfied after migration. By setting the QoS violation device gain to 0, we can suppress the behavior of frequently performing migration that affects SFC traffic transmission. The unit revenue earned at a given time can be calculated as follows:

$$\Delta R^f(t) = \left[\boldsymbol{\mu} \otimes \sum_{n_f \in N_f} C_f(n_f) + \boldsymbol{\eta} \otimes \sum_{l_f \in L_f} C_f(l_f) \right] \cdot \sigma(t) \tag{13}$$

where $\boldsymbol{\mu}$ represents the resource price per node, and $\boldsymbol{\eta}$ represents the resource price per link. $\sigma(t)$ is an indicator variable used to indicate whether the SFC violates QoS requirements at time t, that is:

$$\sigma(t) = \begin{cases} 1, \ \psi_f(t) \le LA_f \\ 0, \ otherwise \end{cases} \tag{14}$$

Let us discuss this expression in the context of different situations:

Case 1: When migration does not occur at time t, the migration indicator variables $m_{n_f}(t)$ and $m_{l_f}(t)$ are both zero vectors, and the total delay $\psi_f(t)$ is solely the waiting delay $\psi_f^0(t)$ of the migrated SFC in the regular operation of the base network. Since the SFC has been embedded, the delay must satisfy the constraint condition that is less than the QoS requirement LA_f (i.e., $\sigma(t) = 1$), so the revenue is calculated normally.

Case 2: When migration occurs at time t, the migration indicator variables $m_{n_f}(t)$ and $m_{l_f}(t)$ are not zero vectors. The total delay $\psi_f(t)$ at this time includes the remapping delay, the waiting delay of the SFC operation, and the rerouting delay in the migration process. It is necessary to determine whether the delay is less than the QoS delay requirement LA_f. The result is $\sigma(t) = 1$ if the requirement is met, and the payoff is calculated normally; it is $\sigma(t) = 0$ if the requirement is not met, in which case the payoff is 0.

As a result, the unified expression for income calculation before and after SFC migration is achieved. When SFC completes its service life cycle, revenue settlement occurs:

$$R^f = \begin{cases} \int_{t_f^a}^{t_f^e} \Delta R^f(t) dt, \ if \ f \ is \ ending \\ 0, \ otherwise \end{cases} \tag{15}$$

It should be noted that this definition also differs from the definition of SFC revenue in other studies. We calculate the revenue of SFC throughout its entire life cycle, rather than just completing the embedding process. If the SFC fails to map after migration due to an operation taking place during migration (i.e., if the SFC ends abnormally), it will be recorded as 0 along with the revenue before migration. This is designed from the perspective that incomplete traffic transmissions will be meaningless. We also hope that the designed algorithm will be integrated into SFC and that the integrity of its traffic will be guaranteed, rather than executing multiple broken traffic transmissions to generate revenue.

During the continuous operation of the network, we prioritize the long-term average revenue of operators:

$$\text{Rev} = \lim_{\tau \to \infty} \frac{1}{\tau} \sum_{f \in \mathcal{F}} R^f \tag{16}$$

3.6. Problem Formulation

The aim of this paper is to maximize the long-term average return for operators. Problem 1 can be expressed as follows:

$$\begin{aligned} \mathcal{P}1: \quad & obj. \max_{\mathbf{X}_f,\mathbf{Y}_f} \text{Rev} = \max_{\mathbf{X}_f,\mathbf{Y}_f} \lim_{t\to\infty} \frac{1}{t} \sum_{f\in\mathcal{F}} R^f \\ s.t. \quad & (1)-(15) \\ & x_f^{n_f,n_S}(t) \in \{0,1\}, y_f^{l_f,l_S}(t) \in \{0,1\} \end{aligned} \tag{17}$$

It should be noted that this is a non-traditional integer programming problem with a time-dependent objective function that needs to be solved quickly, requiring the algorithm to perform well online. At the same time, the decision variables in this problem are discrete 0 and 1 variables related to time, and the constraint conditions are complex and change over time, making this problem strongly discrete and dynamic.

3.7. Hardness Analysis

The challenge of solving problem 1 mainly arises from its extensive solution space. This can be analyzed as follows: the solution space size of node mapping is $|N_f| \times |N_S| \times (t_c - t_a) \times |\mathcal{F}|$, and the solution space of link mapping is $|L_f| \times |L_S| \times (t_c - t_a) \times |\mathcal{F}|$. From the perspective of the online algorithm, only one SFC embedding at a time is considered, simplifying the problem to the knapsack problem with a solution space of $|N_f| \times |N_S| + |L_f| \times |L_S|$. However, the complexity of the problem is further increased by introducing the interaction between time and multiple knapsacks. Rost et al. [26] has proposed that the problem of virtual network embeddings is abstracted as a graph mapping problem and specified using the 3SAT model. The NP-completeness of SFC mapping in a static topology has been proven. Considering the complex constraints, such as dynamic topology and delay, the problem remains NP-hard and non-approximable.

4. Algorithmic Descriptions

4.1. Particle Swarm Optimization for SFC Embedding

Inspired by the regularity of bird flocks' foraging behavior, James Kennedy and Russell Eberhart developed a simplified algorithm model [27], which eventually evolved into particle swarm optimization (PSO) after years of refinement. In order to fully utilize the particle swarm optimization algorithm's ability to search for function extrema in a continuous domain, we consider the probability of mapping a virtual node to each base network node in SFC as the position variable of a particle. Therefore, the dimension of each particle is $D = |N_f| \times |N_S|$, and the position of the i-th particle is:

$$\mathbf{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,D}] = [p_{i,1}(x_f^{n_f^1,n_S^1} = 1), p_{i,2}(x_f^{n_f^1,n_S^2} = 1), \ldots, p_{i,D}(x_f^{|N_f|,|N_S|} = 1)] \tag{18}$$

The range of particle positions is given by $x_{i,d} \in [0,1]$, and the velocity of the i-th particle is:

$$\mathbf{v}_i = [v_{i,1}, v_{i,2}, \ldots, v_{i,D}] \tag{19}$$

We define the mapping relationship between the particle's position and the node when SFC is embedded as:

$$\forall i \in N_f, x_f^{n_f^i,n_S^j} = \begin{cases} 1, & p(x_f^{n_f^i,n_S^j} = 1) \text{ is max}, \forall j \in N_S \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

That is, for the virtual node n_f^i, we will extract the probability vector $\mathbf{x}_f^{n_f^i,*}$ representing the mapping from particles to physical nodes, and select the position with the highest probability to execute the embedding action.

The optimal position sought by the i-th particle in the PSO algorithm (i.e., the individual optimal solution) is:

$$\mathbf{p}_{i,pbest} = [p_{i,1}, p_{i,2}, \ldots, p_{i,D}] \tag{21}$$

The population searches for the optimal position, which is the global optimal solution:

$$\mathbf{p}_{gbest} = [p_{1,gbest}, p_{2,gbest}, \ldots, p_{D,gbest}] \tag{22}$$

The fitness value of the optimal position searched by the i-th particle (i.e., the optimization function f_p) is defined as:

$$f_p = \begin{cases} \left[\boldsymbol{\mu} \otimes \sum\limits_{n_f \in N_f} C_f(n_f) + \boldsymbol{\eta} \otimes \sum\limits_{l_f \in L_f} C_f(l_f)\right] \cdot (\psi_f(t) - LA_f), & \text{if } f \text{ is successfully embedded} \\ \infty, & \text{otherwise} \end{cases} \tag{23}$$

That is to say, if embedding the i-th particle according to its position violates the constraint and results in embedding failure, the fitness value is set to the maximum value. If the embedding is successful, the fitness value is calculated based on the revenue generated by the embedding scheme. Note that we do not directly use Equation (13) as the fitness function because we believe that there is a distinction between the success of the embedding and the transient nature of the QoS violation. There may be a situation where embedding is successful, but the delay in the process of embedding SFC and migration exceeds the QoS requirement. Equation (13) directly calculates this part of the benefit as 0, but further exploration should be encouraged in the PSO algorithm. We calculate the difference between the delay caused by the current mapping and the QoS delay requirement. Then, we combine this with the benefit to incentivize particles to move towards positions in the solution space where the SFC delay is minimized. The lowest fitness value among all particles is taken as the global optimal fitness value f_g.

The velocity update formula for the i-th particle is:

$$\mathbf{v}_i(k+1) = \omega \mathbf{v}_i(k) + c_1 r_1(k)(\mathbf{p}_{i,pbest}(k) - \mathbf{x}_i(k)) + c_2 r_2(k)(\mathbf{p}_{gbest}(k) - \mathbf{x}_i(k)) \tag{24}$$

The position update formula is:

$$\mathbf{x}_i(k+1) = \mathbf{x}_i(k) + \mathbf{v}_i(k+1) \tag{25}$$

Where N represents the number of particles, K represents the number of iterations, and omega is inertia weight. Additionally, c_1 and c_2 are learning factors, while $r_1(k)$ and $r_2(k)$ are random numbers within a specific interval used to enhance the randomness of the search.

The complete flow of the algorithm is shown in Algorithm 1. We give the steps of the proposed algorithm in detail, and we believe that it provides an effective method to solve the problem of online SFC embedding and migration. The computational complexity of a particle swarm optimization algorithm can usually be expressed as $O(NDK)$, where D is the particle dimension, which in our problem is calculated as $N_f N_S$. Meanwhile, the Djikstra algorithm was used to perform routing and link mapping between two VNFs, whose complexity is $O(N_S^2)$. Therefore, the complexity of the algorithm proposed in this paper is $O(NN_f N_S K + N_f N_S^2)$.

Algorithm 1 Particle Swarm Optimization Algorithm for SFC Embedding

```
Input: Current substrate network $G_S = (N_S, L_S)$; Current SFC request $f$; Dimension of
    particle position $dim$; Max iterations $maxIter$; Number of particles $pop$.
Output: SFC embedding scheme node mapping $\mathbf{X}_f$ and link mapping $\mathbf{Y}_f$, $flag$ indicating
    whether embedding succeeds.
 1: for all each particle $i$ do
 2:     for all each dimension $d$ do
 3:         Randomly initialize the particle position $x_{i,d}$ in the range [0,1];
 4:         Randomly initialize the particle velocity $v_{i,d}$ in the range [0,1];
 5:     end for
 6: end for
 7: Initialize current iteration number $iter = 1$;
 8: while $iter < maxIter$ do
 9:     for all each particle $i$ do
10:         Calculate fitness value $f_p$ with Equation (23);
11:         if the fitness $f_p$ is less than the particle historical optimum $p^k_{i,pbest}$ then
12:             Set the current fitness $f_p$ to $p^k_{i,pbest}$;
13:         end if
14:     end for
15:     Select the smallest historical optimal fitness value among all particles as the global
    optimal fitness value $p^k_{gbest}$;
16:     for all each particle $i$ do
17:         Calculate velocity according to the Equation (24);
18:         Update particle position according to the Equation (25);
19:     end for
20:     $iter = iter + 1$;
21: end while
22: if the fitness $f_p$ is the limiting value $Inf$ then
23:     $\mathbf{X}_f = None, \mathbf{Y}_f = None, flag = False$;
24: else
25:     Convert the global optimum fitness particle position $\mathbf{x}_i$ to a node embedding scheme
    $\mathbf{X}_f$ with Equation (20);
26:     Solve the shortest path between embedded nodes with Dijkstra algorithm, and the
    link embedding scheme $\mathbf{Y}_f$ is obtained;
27:     $flag = True$
28: end if
29: return $\mathbf{X}_f, \mathbf{Y}_f, flag$;
```

4.2. Analysis of Convergence of Algorithm

We conduct a dynamic analysis of the particle's trajectory during the algorithm iteration, and the velocity update is expressed as:

$$\mathbf{v}(k+1) = \omega\mathbf{v}(k) + c_1 r_1(k)(\mathbf{p}_{pbest}(k) - \mathbf{x}(k)) + c_2 r_2(k)(\mathbf{p}_{gbest}(k) - \mathbf{x}(k)) \tag{26}$$

Position updates are indicated by:

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \mathbf{v}(k+1) \tag{27}$$

Let $\varphi_1(k) = c_1 r_1(k)$, $\varphi_2(k) = c_2 r_2(k)$, and $\varphi(k) = \varphi_1(k) + \varphi_2(k)$.
Thus, the updates for velocity and position can be rewritten as:

$$\mathbf{v}(k+1) = \omega\mathbf{v}(k) - \varphi(k)\mathbf{x}(k) + \varphi_1(k)\mathbf{p}_{pbest}(k) + \varphi_2(k)\mathbf{p}_{gbest}(k) \tag{28}$$

$$\mathbf{x}(k+1) = \omega\mathbf{v}(k) + (1 - \varphi(k))\mathbf{x}(k) + \varphi_1(k)\mathbf{p}_{pbest}(k) + \varphi_2(k)\mathbf{p}_{gbest}(k) \tag{29}$$

The matrix form can be expressed as:

$$\begin{bmatrix} \mathbf{v}(k+1) \\ \mathbf{x}(k+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega} & -\boldsymbol{\varphi}(k) \\ \boldsymbol{\omega} & \mathbf{I}-\boldsymbol{\varphi}(k) \end{bmatrix} \begin{bmatrix} \mathbf{v}(k) \\ \mathbf{x}(k) \end{bmatrix} + \begin{bmatrix} \boldsymbol{\varphi}_1(k) & \boldsymbol{\varphi}_2(k) \\ \boldsymbol{\varphi}_1(k) & \boldsymbol{\varphi}_2(k) \end{bmatrix} \begin{bmatrix} \mathbf{p}_{pbest}(k) \\ \mathbf{p}_{gbest}(k) \end{bmatrix} \tag{30}$$

According to the hypothesis in [28], the position corresponding to the current optimal fitness value of the particle is the global optimal position, and it has become a fixed value $\mathbf{p}_{pbest}(k) = \mathbf{p}_{gbest}(k) = \mathbf{p}$ in the long-term iteration. In this case, the dynamics model of PSO is rewritten as:

$$\begin{bmatrix} \mathbf{v}(k+1) \\ \mathbf{x}(k+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega} & -\boldsymbol{\varphi}(k) \\ \boldsymbol{\omega} & \mathbf{I}-\boldsymbol{\varphi}(k) \end{bmatrix} \begin{bmatrix} \mathbf{v}(k) \\ \mathbf{x}(k) \end{bmatrix} + \begin{bmatrix} \boldsymbol{\varphi}(k)\mathbf{p} \\ \boldsymbol{\varphi}(k)\mathbf{p} \end{bmatrix} \tag{31}$$

where $\mathbf{v}(k+1) = [v_1(k+1), v_2(k+1), \dots v_D(k+1)]^T$, $\mathbf{v}(k) = [v_1(k), v_2(k), \dots v_D(k)]^T$ is the velocity vector of the particle during iteration. $\mathbf{x}(k+1) = [x_1(k+1), x_2(k+1), \dots x_D(k+1)]^T$, and $\mathbf{x}(k) = [x_1(k), x_2(k), \dots x_D(k)]^T$ are the position vectors of particles in the iterative process. The parameters in the original formula are expressed in matrix form as follows:

$$\boldsymbol{\omega} = \begin{bmatrix} \omega & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \omega \end{bmatrix}_{D\times D}, \boldsymbol{\varphi}(k) = \begin{bmatrix} \varphi(k) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \varphi(k) \end{bmatrix}_{D\times D}, \mathbf{p} = \begin{bmatrix} p_1 \\ \vdots \\ p_D \end{bmatrix} \tag{32}$$

We can think of this particle as a dynamic system whose coefficient matrix is:

$$\mathbf{A}(k) = \begin{bmatrix} \boldsymbol{\omega} & -\boldsymbol{\varphi}(k) \\ \boldsymbol{\omega} & \mathbf{I}-\boldsymbol{\varphi}(k) \end{bmatrix} \tag{33}$$

The stability criterion of the system (i.e., ensuring that the eigenvalues of the coefficient matrix $\mathbf{A}(k)$ are less than 1) is used for analysis. Find the eigenvalues of the block matrix $\mathbf{A}(k)$, that is, solve the equation $\det(sI - A(k)) = 0$.

$$\begin{aligned} \det(s\mathbf{I} - \mathbf{A}(k)) &= \det\begin{bmatrix} s\mathbf{I}-\boldsymbol{\omega} & \boldsymbol{\varphi}(k) \\ -\boldsymbol{\omega} & s\mathbf{I}-\mathbf{I}+\boldsymbol{\varphi}(k) \end{bmatrix} \\ &= \det[s\mathbf{I}-\boldsymbol{\omega}] \times \det[s\mathbf{I}-\mathbf{I}+\boldsymbol{\varphi}(k) - (-\boldsymbol{\omega})(s\mathbf{I}-\boldsymbol{\omega})^{-1}\boldsymbol{\varphi}(k)] \\ &= (s-\omega)^D(s+\varphi(k)-1+\frac{\omega\varphi(k)}{s-\omega})^D \\ &= [s^2 + s(\varphi(k)-\omega-1)+\omega]^D \\ &= 0 \end{aligned} \tag{34}$$

For $\varphi(k) - \omega - 1 = -u(k)$, the equation can be rewritten as:

$$s^2 - u(k)s + \omega = 0 \tag{35}$$

The characteristic roots are $s_1 = \frac{u(k)+\sqrt{u^2(k)-4\omega}}{2}$ and $s_2 = \frac{u(k)-\sqrt{u^2(k)-4\omega}}{2}$, respectively, under the condition $|s| < 1$.

The conditions for stable convergence are:

$$\begin{cases} 1+\omega > |u(k)| \\ |\omega| < 1 \end{cases} \tag{36}$$

where $u(k) = -\varphi(k) + \omega + 1$. The equation can be solved as follows:

$$\begin{cases} 0 < \varphi(k) = \varphi_1(k) + \varphi_2(k) = c_1 r_1(k) + c_2 r_2(k) < 2(\omega+1) \\ |\omega| < 1 \end{cases} \tag{37}$$

Our particle swarm optimization algorithm uses the parameters that satisfy this condition to ensure the stability of convergence. Although the parameter selection conditions that make the algorithm in this paper converge can be analyzed from the above content, an important decision of whether the particle swarm converges still depends on the fitness function with Equation (23). The fitness function determines the movement direction of the particle and affects the process of particle searching for the optimal solution, while the design of learning factors and weights ensures the asymptotic stability of the particle adjustment process. The complexity of the fitness function lies in its discreteness. Although we make the particle variables continuous by mapping probabilities, the function values are still discrete, and the direct reason for this is the discrete network structure. We have not yet solved how to prove the discreteness of the algorithm directly from the fitness function, but only hope that the embedding strategy of SFC under the learning factors and weights of PSO is not so sloppy.

5. Performance Evaluation

5.1. Simulation Model

In order to validate the effectiveness and performance of the proposed resource-aware SFC embedding and migration algorithm for UAV dynamic networks, we developed a simulation framework for SFC embedding based on MANO architecture using Python 3.9 (https://gitee.com/WangXi_Chn/mini_sfc, accessed on 3 February 2024), with reference to [29], as shown in Figure 2.

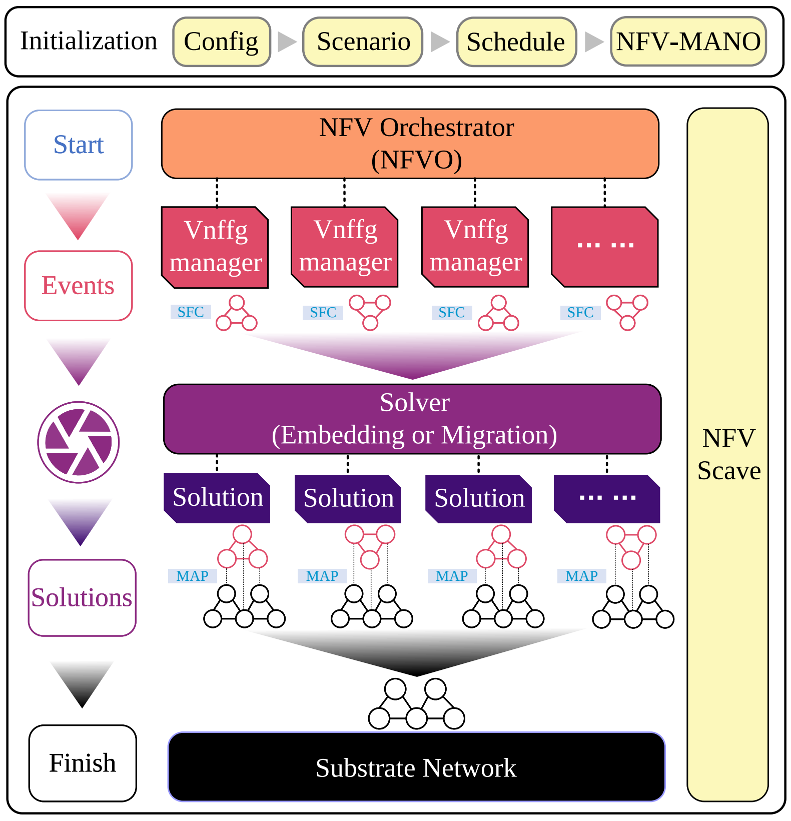

Figure 2. SFC embedding and migration simulation framework based on MANO.

Designed and proposed by the European Telecommunications Standards Institute (ETSI), the MANO architecture provides a well-defined framework for NFV that is particularly adept at handling the intricate dependencies and operational dynamics of network services. The MANO architecture's ability to interface with various virtual network functional infrastructure (VNFI) domains makes it better suited for the heterogeneous environment of UAV IoT, where interoperability and integration with different network elements and services is critical. We designed the simulation model of SFC embedding and migration algorithm with reference to MANO architecture, so that it has certain standardization, reproducibility, and replicability.

The simulation framework mainly comprises the following components:

- The initialization model is used to configure and schedule events in simulation scenarios, initialize all NFV-MANO modules, and make preparations for the simulation.
- The virtual network function orchestrator (VNFO) model is used to orchestrate and manage the SFC. This includes tasks such as creating, embedding, migrating, and destroying the SFC, as well as calculating the revenue model of the SFC and optimizing the objective function through a solver.
- The virtual network function forwarding graph (VNFFG) model describes the topology, functional requirements, and quality of service of the SFC, as well as the life cycle status of the SFC, including creation, embedding, running, migration, and destruction.
- The NFV Scave model is used to monitor and maintain SFC, which involves collecting and analyzing SFC operating status, performance indicators, energy consumption, and resources. It also includes making judgments and executing SFC migration trigger conditions.

The simulation framework utilizes a discrete event-driven approach, meaning it advances the simulation clock and processes events based on their occurrence time and type. Events are primarily categorized into endogenous events and exogenous events. Endogenous events are those caused by state changes resulting from internal activities of the system, such as the creation, embedding, migration, and destruction of SFC, while exogenous events are those caused by external factors affecting the system, such as the movement, communication, and energy consumption of UAVs. The timing and nature of events are determined by the relevant probability distribution or rules, and the processing of events is carried out using the corresponding model or algorithm. The execution flow of the simulation framework is depicted in Figure 3.

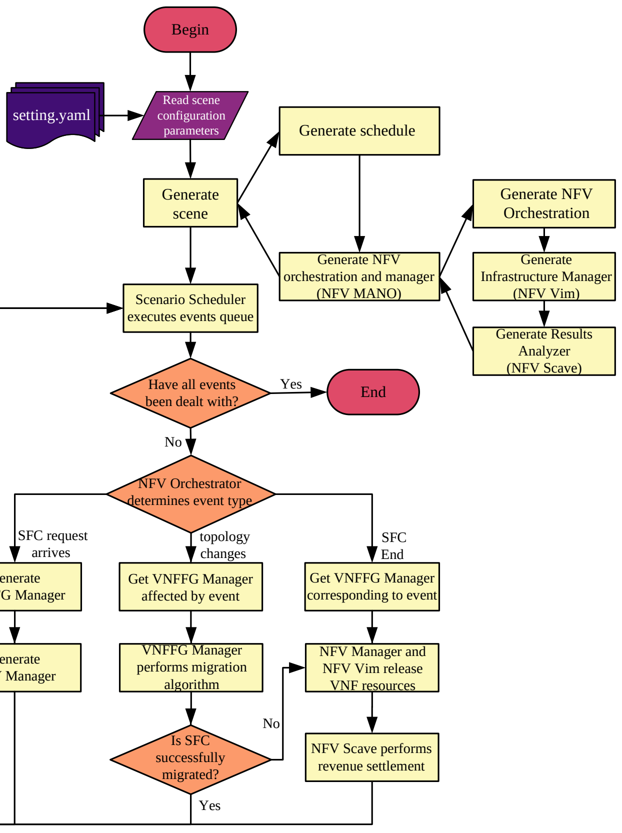
Begin
setting.yaml
Read scene configuration parameters
Generate schedule
Generate scene
Generate NFV Orchestration
Generate NFV orchestration and manager (NFV MANO)
Generate Infrastructure Manager (NFV Vim)
Generate Results Analyzer (NFV Scave)
Scenario Scheduler executes events queue
Have all events been dealt with?
Yes
End
No
NFV Orchestrator determines event type
SFC request arrives
topology changes
SFC End
Generate VNFFG Manager
Get VNFFG Manager affected by event
Get VNFFG Manager corresponding to event
Generate NFV Manager
VNFFG Manager performs migration algorithm
NFV Manager and NFV Vim release VNF resources
No
Is SFC successfully migrated?
NFV Scave performs revenue settlement
Yes

Figure 3. Simulation model workflow.

5.2. Simulation Setting

In this section, we will showcase the performance of the PSO algorithm for SFC deployment in the emulator. Since there is currently no online algorithm for income calculation based on long-term operation, this paper chooses to compare it with random and greedy strategies. Random strategy refers to NFVO randomly selecting physical nodes in the current base network for VNF embedding after receiving SFC deployment requests from the environment and SFC migration due to topology changes. The greedy strategy involves selecting physical nodes with ample remaining resources to embed as much as possible, and then using the Dijkstra algorithm to sequentially route each node for embedding virtual links.

In the simulation, we consider a UAV network with 30 nodes as the base network, and the network topology is randomly generated using the Waxman model. According to the literature [25], we set the capacity of CPU, RAM, and DISK resources of each site in the network to 20 units and set the battery capacity to 200 units. The bandwidth resource on the physical link is 200 units. For each SFC, its source and destination sites are randomly selected from the network topology, and the number of VNFs ranges from 8 to 10. Each VNF instance requires a uniformly distributed random number of CPU, RAM, and DISK resources (1–10). The requested virtual link bandwidth resource is a uniformly distributed random number between 10 and 100. The life cycle of each SFC follows a uniform distribution of (5,10) time units, and the delay requirement in QoS consists of uniformly distributed random numbers in the range of [0.2, 0.3]. We generate multiple SFC requests in the scheduler of the simulated event, and the SFC requests arrive at the system in a Poisson process sequence. At the same time, in order to simulate topological dynamics while maintaining network connectivity, the physical links are randomly dismantled and rebuilt at intervals. When the SFC's life cycle ends, NFVO removes it from the base network to indicate that its task has been completed. Set the time cost for updating a forwarding rule on the switch when the VNF is remapped between two nodes to 0.01 unit. The minimum wait time in the delay list for applying for resources on a node is 0.01 unit. Each experiment was repeated 10 times, and the results were averaged to minimize the impact of chance.

5.3. Network Resource Consumption

To evaluate the network resource awareness of the PSO, we compared it to the baseline algorithm using the simulation settings mentioned above. Figure 4 illustrates the fluctuations in the remaining CPU, RAM, DISK, and ENG resources during operation in the same network scenario. We can observe from the utilization of CPU, RAM, and DISK leased resources represented by Figure 4a–c that the PSO algorithm can effectively utilize the available resources in the network. During the initial phase of scenario operation, the SFC completion rate is high, leading to a rapid increase in resource utilization, exceeding 50%. In contrast, the baseline algorithm remains below this level for an extended period. In the later stages of the scenario operation, resource utilization under the PSO algorithm gradually decreases. This is mainly influenced by the remaining power resources on the UAV node, as depicted in the ENG consumption resource utilization represented by Figure 4d. After analyzing all the events, the utilization of network power resources under the PSO algorithm reaches 79%, while the greedy algorithm and random algorithm in the baseline algorithm correspond to 58% and 10%, respectively. It is evident that deploying SFC through the PSO algorithm can improve network resource awareness and achieve optimal resource utilization.

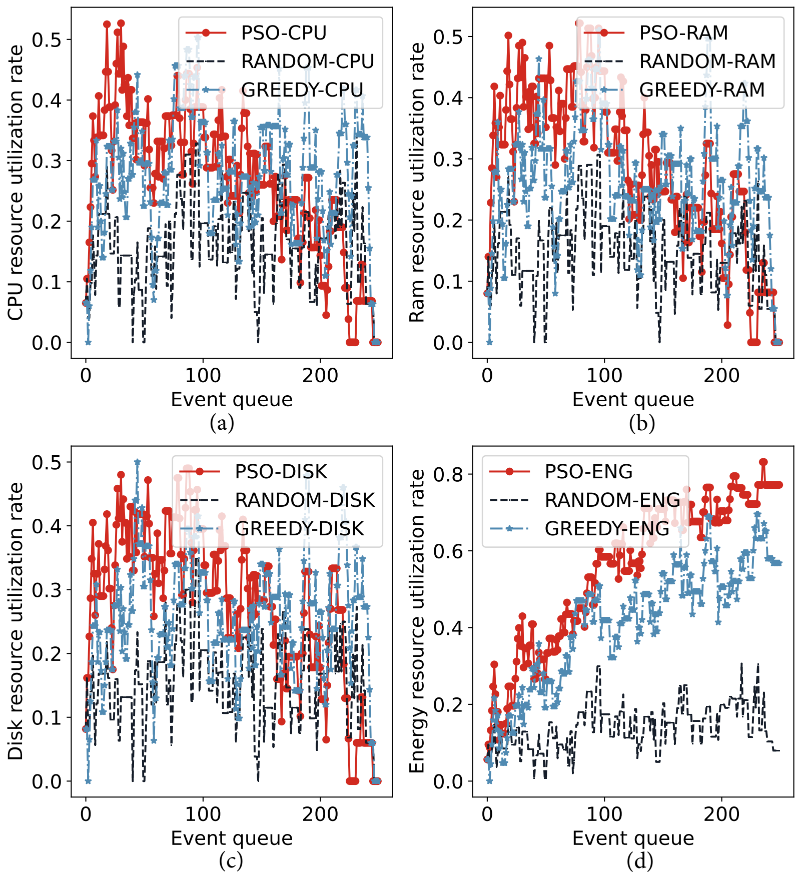

Figure 4. Dynamic diagram illustrating changes in network resource utilization: (**a**) CPU resource; (**b**) RAM resource; (**c**) DISK resource; (**d**) energy resource.

5.4. Impact of Topological Change

To assess the algorithm's performance in a UAV network with dynamic topological changes, we introduced various topological change events at different times in the base network during the operation process. We then compared and observed the completion rate of SFC, total revenue, and long-term average revenue during the experiment. The result is depicted in Figure 5. Given that the number of network topology changes during operation falls within the range of 10 to 50, the experiment is repeated 5 times. In Figure 5a, it can be seen that the PSO-based embedding algorithm has a significantly higher SFC completion rate than the baseline algorithm by approximately 50%. It is also less affected by the severity of topological changes, whereas the greedy algorithm and random algorithm are only 37% and 10%. Figure 5b,c demonstrate that the PSO algorithm also outperforms the two baseline algorithms in terms of the total and long-term average benefits of the running process. However, as the complexity of the topology changes increases, the income situation of PSO gradually deteriorates to the level of the greedy algorithm. This demonstrates that although SFC can maintain a high completion rate and provide an embedding scheme that meets the constraint conditions under the action of the PSO algorithm, the frequent rerouting and remapping delays in the migration process prevent it from obtaining benefits normally.

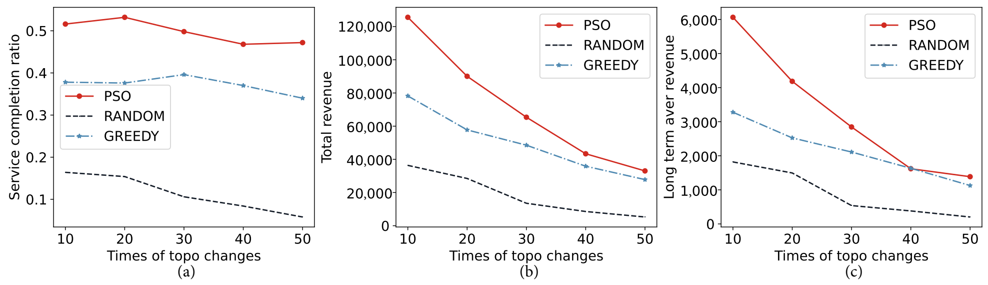

Figure 5. Comparison of SFC algorithm embedding performance under different frequencies of topological changes: (**a**) comparison of SFC completion rates; (**b**) comparison of total system revenue; (**c**) comparison of long-term average revenue.

5.5. Impact of Workload

To investigate the influence of service load on the SFC embedding algorithm, we varied the SFC arrival rates for our experiments, and the results are depicted in the figure. Figure 6a illustrates the completion of the SFC as the service load gradually increases. It is evident that the PSO algorithm can maintain a stable level, while the greedy algorithm and random algorithm gradually deteriorate. This gap gradually expands with an increase in load. This phenomenon can also be observed in the total returns and long-term average returns depicted in Figure 6b,c. This result demonstrates that the PSO algorithm is capable of thoroughly exploring the solution space of the problem and offering a viable placement or migration plan, even when there are more SFC services in the network and fewer remaining resources. This also exemplifies its resource-aware ability. It should be noted that under optimal conditions, the arrival rate of SFC ultimately impacts the fluctuation of resource occupancy in the network. Hence, apart from the embedding algorithm, the bottleneck also lies in the distribution of network resources, which can result in significant random errors in experimental outcomes.

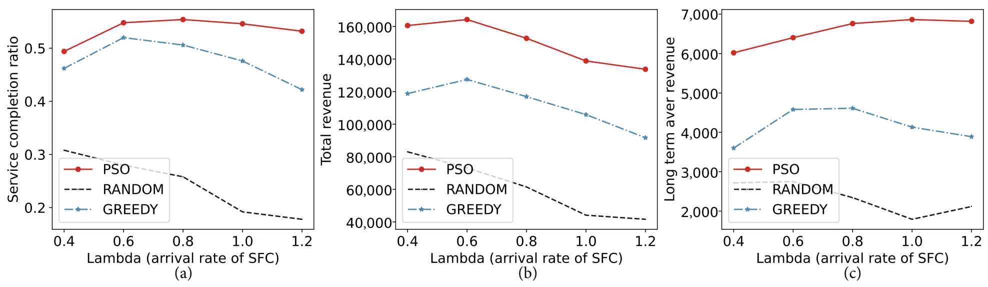

Figure 6. Comparison of SFC algorithm embedding performance under different arrival rates of SFCs: (**a**) comparison of SFC completion rates; (**b**) comparison of total system revenue; (**c**) comparison of long-term average revenue.

5.6. Impact on Volume of Services

In the experiment, we are investigating the network's capacity by using different amounts of SFC. Specifically, we gradually increase the amount of network resources requested to explore the network's ultimate capacity. Leased resources, such as CPU, RAM, and DISK, have recyclable characteristics that result in minimal impact on the increase in the number of SFCs when the arrival rate and life cycle remain unchanged. As a result, revenue increases steadily over time. However, for consumable resources such as ENG, once the energy is depleted, the network will no longer be able to provide resources for any SFCs. As shown in Figure 7a, as the number of SFCs increases, the PSO algorithm gradually reaches the bottleneck of network resources, and the completion rate approaches that of the greedy algorithm. It can also be calculated that the maximum number of SFCs that can provide resources in the network is approximately 30. It can also be observed from the total and long-term returns in Figure 7b,c that the yield curve reaches its limit when energy resources are near exhaustion.

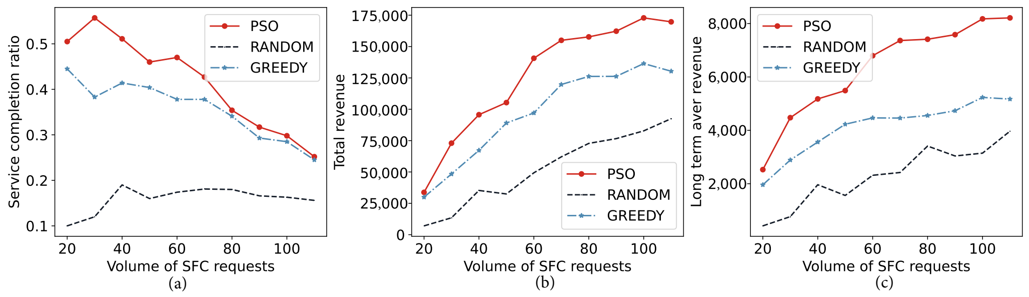

Figure 7. Comparison of SFC algorithm embedding performance under different volumes of SFCs: (**a**) comparison of SFC completion rates; (**b**) comparison of total system revenue; (**c**) comparison of long-term average revenue.

6. Discussion and Future Work

This paper has presented an in-depth examination of the embedding and migration of service function chaining (SFC) within a dynamic network topology, specifically a UAV network. The study focused on the comprehensive process of calculating the benefits of SFC, from its deployment to maintenance throughout its lifecycle, with the long-term average benefit as the optimization goal. The problem was formulated as an integer programming problem and addressed with a proposed SFC embedding algorithm based on the particle swarm optimization (PSO) algorithm. This algorithm provides an SFC embedding and migration scheme with reduced time costs. Experimental results demonstrated the algorithm's ability to achieve network resource perception by leveraging the exploratory capabilities of particles, thereby efficiently utilizing network resources across varying loads, from low to high.

However, we acknowledges the limitation of the local optimal solution bottleneck of the PSO algorithm as a heuristic algorithm. Moreover, the performance of optimized deployment under given simulation settings is not presented in this paper and then compared with the used algorithm to reflect the degree of optimality of the algorithm. This is mainly because using the existing network modeling methods in dynamic scenarios, it is extremely complicated to solve the optimal deployment scheme, and this complexity is mainly related to the sequential decisions brought by considering the time factor. As far as we know, the solution of the optimal SFC embedding mode in a dynamic network environment has not been studied in the existing research. In fact, our verification experiment on network resource consumption can also demonstrate the optimality of this algorithm to a certain extent, because at the end of the simulation, the battery resource utilization rate of the UAV swarm has reached about 80%, and it is already very difficult to further deploy SFC to the network under the constraint of energy consumption. We do not yet have an offline algorithm that can tell us what the ideal limit of resource utilization is. In our recent research, modeling dynamic networks using a time extend graph (TEG) combined with maximum flow theory is a possible way to achieve this goal, which is one of our future research plans.

Future research is recommended to propose an online algorithm with similar agility but improved exploration performance, further optimizing the utilization of network resources in conjunction with the statistical law of SFC service arrival and SFC context. This approach will enable progress towards the global optimal solution. Intelligent algorithms, such as those represented by deep reinforcement learning, may be one of the potential solutions to this challenge. This study's findings and proposed algorithm have significant implications for the field of dynamic network topology, particularly in the context of UAV networks.

Author Contributions: Conceptualization, X.W.; methodology, X.W. and S.S.; validation, S.S.; writing—original, X.W.; writing—review and editing, S.S. and C.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work has received support from the National Natural Science Foundation of China under Grant 62171158 and was also supported by the Fundamental Research Funds for the Central Universities under Grant 2242022k60006.

Data Availability Statement: The data are inconvenient to directly disclose. The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Chaumette, S.; Kim, J.H.; Namuduri, K.; Sterbenz, J.P.G. *UAV Networks and Communications*; Cambridge University Press: Cambridge, UK, 2017.
2. Zeng, Y.; Xu, X.; Jin, S.; Zhang, R. Simultaneous Navigation and Radio Mapping for Cellular-Connected UAV With Deep Reinforcement Learning. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 4205–4220. [CrossRef]
3. Qin, Y.; Guo, D.; Luo, L.; Zhang, J.; Xu, M. Service function chain migration with the long-term budget in dynamic networks. *Comput. Netw.* **2023**, *223*, 109563. [CrossRef]
4. Attaoui, W.; Sabir, E.; Elbiaze, H.; Guizani, M. VNF and CNF Placement in 5G: Recent Advances and Future Trends. *IEEE Trans. Netw. Serv. Manag.* **2023**, *20*, 4698–4733. [CrossRef]
5. Wijethilaka, S.; Liyanage, M. Survey on Network Slicing for Internet of Things Realization in 5G Networks. *IEEE Commun. Surv. Tutorials* **2021**, *23*, 957–994. [CrossRef]
6. Wu, W.; Zhou, C.; Li, M.; Wu, H.; Zhou, H.; Zhang, N.; Shen, X.S.; Zhuang, W. AI-Native Network Slicing for 6G Networks. *IEEE WIreless Commun.* **2022**, *29*, 96–103. [CrossRef]
7. Beck, M.T.; Fischer, A.; de Meer, H.; Botero, J.F.; Hesselbach, X. A distributed, parallel, and generic virtual network embedding framework. In Proceedings of the 2013 IEEE International Conference on Communications (ICC), Budapest, Hungary, 9–12 June 2013; pp. 3471–3475.
8. Beck, M.T.; Botero, J.F. Coordinated Allocation of Service Function Chains. In Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM), San Diego, CA, USA, 6–10 December 2015. [CrossRef]
9. Tomassilli, A.; Giroire, F.; Huin, N.; Pérennes, S. Provably Efficient Algorithms for Placement of Service Function Chains with Ordering Constraints. In Proceedings of the IEEE INFOCOM 2018—IEEE Conference on Computer Communications, Honolulu, HI, USA, 16–19 April 2018; pp. 774–782. [CrossRef]
10. Sallam, G.; Gupta, G.R.; Li, B.; Ji, B. Shortest path and maximum flow problems under service function chaining constraints. In Proceedings of the IEEE Infocom 2018—IEEE Conference on Computer Communications, Honolulu, HI, USA, 16–19 April 2018; pp. 2132–2140.
11. Jin, P.; Fei, X.; Zhang, Q.; Liu, F.; Li, B. Latency-aware VNF chain deployment with efficient resource reuse at network edge. In Proceedings of the IEEE INFOCOM 2020—IEEE Conference on Computer Communications, Toronto, ON, Canada, 6–9 July 2020; pp. 267–276.
12. Zheng, D.; Peng, C.; Liao, X.; Cao, X. Toward optimal hybrid service function chain embedding in multiaccess edge computing. *IEEE Internet Things J.* **2019**, *7*, 6035–6045. [CrossRef]
13. Zheng, D.; Peng, C.; Liao, X.; Tian, L.; Luo, G.; Cao, X. Towards latency optimization in hybrid service function chain composition and embedding. In Proceedings of the IEEE INFOCOM 2020—IEEE Conference on Computer Communications, Toronto, ON, Canada, 6–9 July 2020; pp. 1539–1548.
14. Dimolitsas, I.; Dechouniotis, D.; Papavassiliou, S. Time-efficient distributed virtual network embedding for round-trip delay minimization. *J. Netw. Comput. Appl.* **2023**, *217*, 103691. [CrossRef]
15. Vidal, I.; Nogales, B.; Valera, F.; Gonzalez, L.F.; Sanchez-Aguero, V.; Jacob, E.; Cervelló-Pastor, C. A multi-site NFV testbed for experimentation with SUAV-based 5G vertical services. *IEEE Access* **2020**, *8*, 111522–111535. [CrossRef]
16. Carpio, F.; Bziuk, W.; Jukan, A. On optimal placement of hybrid service function chains (SFCs) of virtual machines and containers in a generic edge-cloud continuum. *arXiv* **2020**, arXiv:2007.04151.
17. Carpio, F.; Bziuk, W.; Jukan, A. Scaling migrations and replications of virtual network functions based on network traffic forecasting. *Comput. Netw.* **2022**, *203*, 108582. [CrossRef]
18. Rui, L.; Chen, X.; Gao, Z.; Li, W.; Qiu, X.; Meng, L. Petri Net-Based Reliability Assessment and Migration Optimization Strategy of SFC. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 167–181. [CrossRef]
19. Bai, J.; Chang, X.; Rodríguez, R.J.; Trivedi, K.S.; Li, S. Towards uav-based mec service chain resilience evaluation: A quantitative modeling approach. *IEEE Trans. Veh. Technol.* **2023**, *72*, 5181–5194. [CrossRef]
20. Hu, Y.; Min, G.; Li, J.; Li, Z.; Cai, Z.; Zhang, J. VNF Migration in Digital Twin Network for NFV Environment. *Electronics* **2023**, *12*, 4324. [CrossRef]
21. Boutin, E.; Ekanayake, J.; Lin, W.; Shi, B.; Zhou, J.; Qian, Z.; Wu, M.; Zhou, L. Apollo: Scalable and coordinated scheduling for cloud-scale computing. In Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation, Broomfield, CO, USA, 6–8 October 2014; pp. 285–300.
22. Schneider, S.; Sharma, A.; Karl, H.; Wehrheim, H. Specifying and Analyzing Virtual Network Services Using Queuing Petri Nets. In Proceedings of the 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Arlington, VA, USA, 8–12 April 2019; pp. 116–124.

23. Li, J.; Wang, R.; Wang, K. Service Function Chaining in Industrial Internet of Things With Edge Intelligence: A Natural Actor-Critic Approach. *IEEE Trans. Ind. Inform.* **2023**, *19*, 491–502. [CrossRef]
24. Su, S.; Zhang, Z.; Liu, A.X.; Cheng, X.; Wang, Y.; Zhao, X. Energy-Aware Virtual Network Embedding. *IEEE/ACM Trans. Netw.* **2014**, *22*, 1607–1620. [CrossRef]
25. Wang, T.; Fan, Q.; Li, X.; Zhang, X.; Xiong, Q.; Fu, S.; Gao, M. DRL-SFCP: Adaptive Service Function Chains Placement with Deep Reinforcement Learning. In Proceedings of the ICC 2021—IEEE International Conference on Communications, Montreal, QC, Canada, 14–23 June 2021; pp. 1–6. [CrossRef]
26. Rost, M.; Schmid, S. On the Hardness and Inapproximability of Virtual Network Embeddings. *IEEE/ACM Trans. Netw.* **2020**, *28*, 791–803. [CrossRef]
27. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948. [CrossRef]
28. Clerc, M.; Kennedy, J. The particle swarm—Explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.* **2002**, *6*, 58–73. [CrossRef]
29. Castillo-Lema, J.; Venâncio Neto, A.; de Oliveira, F.; Takeo Kofuji, S. Mininet-NFV: Evolving Mininet with OASIS TOSCA NVF profiles Towards Reproducible NFV Prototyping. In Proceedings of the 2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France, 24–28 June 2019; pp. 506–512. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.