An Efficient Service Function Chains Orchestration Algorithm for Mobile Edge Computing

Xiulei Wang[1], Bo Xu[1*], and Fenglin Jin[1]
[1] Institute of Command and Control Engineering, Peoples Liberation Army Engineering University, Nanjing, 210007, China
[e-mail: xiuleiwang1988@126.com; xubo820@163.com; 18951003070@189.cn]
[*]Corresponding author: Bo Xu

Received July 23, 2021; revised October 8, 2021; accepted November 25, 2021; published December 31, 2021

Abstract

The dynamic network state and the mobility of the terminals make the service function chain (SFC) orchestration mechanisms based on static and deterministic assumptions hard to be applied in SDN/NFV mobile edge computing networks. Designing dynamic and online SFC orchestration mechanism can greatly improve the execution efficiency of compute-intensive and resource-hungry applications in mobile edge computing networks. In order to increase the overall profit of service provider and reduce the resource cost, the system running time is divided into a sequence of time slots and a dynamic orchestration scheme based on an improved column generation algorithm is proposed in each slot. Firstly, the SFC dynamic orchestration problem is formulated as an integer linear programming (ILP) model based on layered graph. Then, in order to reduce the computation costs, a column generation model is used to simplify the ILP model. Finally, a two-stage heuristic algorithm based on greedy strategy is proposed. Four metrics are defined and the performance of the proposed algorithm is evaluated based on simulation. The results show that our proposal significantly provides more than 30% reduction of run time and about 12% improvement in service deployment success ratio compared to the Viterbi algorithm based mechanism.

Keywords: Mobile Edge Computing, Internet of Things, Service Function Chain, Column Generation, Network Function Virtualization.

http://doi.org/10.3837/tiis.2021.12.006 **ISSN : 1976-7277**

1. Introduction

In recent years, a variety of compute-intensive and resource-hungry applications such as autonomous driving, face recognition, intelligent manufacturing and so on are applied in the mobile networks [1][2]. In order to support these high QoS requirements applications, more agile and efficient technologies, such as Software Defined Network (SDN), Network Function Virtualization (NFV) and Mobile Edge Computing (MEC) are proposed [3][4]. The network services can be provided in software-enable network functions or elements by leveraging NFV. The SDN supplies a centralized network control paradigm and flexibly customized traffic routing and steer the traffic to traverse through VNFs in order. The MEC is proposed to bring computation resource closer to end users by installing small resource-limited cloud at network edge, so as to provide delay-guaranteed service to end users. The service function chain (SFC) is a typical application which takes advantage of NFV, SDN to improve the performance of complicated network services in MEC [5][30].

In SFC, a complex task is decomposed into a sequence of virtual network functions (VNF) or micro-services based on NFV technology. Then each VNF of the SFC is mapped into the underlying network node and the data flow is routed between those nodes. The deployment of SFC request includes how to map the VNFs to the underlying network nodes and how to route the data flow between the VNF nodes in order. For most of static SFC orchestration mechanisms, the fluctuation of the traffic and the mobility of the users will make the VNF deployment and the path among VNFs sub-optimal. Therefore, when the end user moves, the service needs to be rearranged [7].

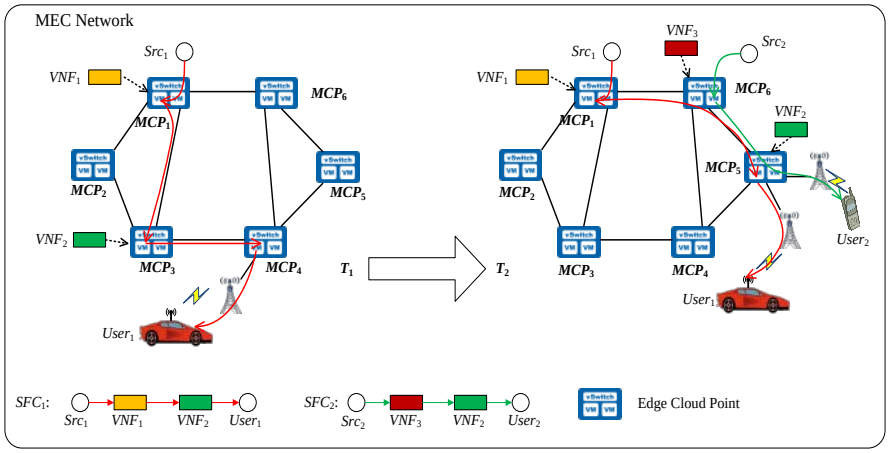


Fig. 1. SFC dynamic orchestration in mobile edge computing networks.

As an illustrative example, the dynamic SFC deployment and adjustment scenario in MEC is shown in **Fig. 1**. In this scenario, an access-edge cloud is assigned to the base station (BS). The resource of MEC server is dynamically allocated to instantiate VNFs. A VNF can process the traffic from the other VNFs [8]. Here we all each MEC server as Edge Cloud Point (MCP). At T_1 the in-service SFC request from $User_1$ access the network from MCP_4 and takes an SFC as $Src_1 \rightarrow VNF_1 \rightarrow VNF_2 \rightarrow User_1$. To deploy the SFC of $User_1$, a VNF_1 and a VNF_2 is deployed on MCP_1 and MCP_3. With the time goes by, at T_2, a new user $User_2$ joints at MCP_5 to request an SFC as $Src_2 \rightarrow VNF_3 \rightarrow VNF_2 \rightarrow User_2$ and the in-server $User_1$ has roamed to

MCP_5 but its SFC stays unchanged.

In this paper, the impact of dynamic network states and the mobility of endpoints on SFCs orchestration is investigated. Constrained by the MCP computing resources, link bandwidth and user's QoS requests, service provider will jointly consider the deployment of new request SFC_2 and the adjustment of SFC_1 simultaneously, aiming at reducing resource cost and increasing benefit of service provider. The SFC dynamic orchestration problem is formulated as an integer linear programming model based on the layered graph by considering the network resource constraints, traffic routing and VNF sequence constraint. The problem is proved to be NP-hard. Then a greedy idea based algorithm based on column generation [9] is proposed to reduce the time cost and obtain the approximate optimal solution. The main work of this paper is summarized as follows.

1. Based on an SFC dynamic deployment scenario, the SFC dynamic deployment problem is defined and modeled. Aiming at minimize the overall resources cost and maximize the profit of service provider; we divide the running process of the system into different time slots. In each time slot, we formulate the SFC dynamic deployment problems as an ILP model based on layered graph.

2. In order to reduce the number of variables and reduce the space of feasible solutions, we treat the feasible path of each SFC request as a column of the column generation model, and modify the ILP model with the column generation model. Based on the framework of column generation algorithm, a two-stage SFC dynamic scheduling heuristic algorithm is designed by greedy idea, which greatly reduces the time complexity.

3. Four performance metrics are defined and some numerical experiments are designed to evaluate the performance of the mechanism. The efficient of the algorithm is proved.

The paper is organized as follows. In Section 2, we review the previous works related to SFC orchestration. In Section 3, the problem is described and definitely defined. In Section 4, we first formulate the problem as an Integer Linear Program (ILP) model and then a CG decomposition model is used to reformulate the problem. In Section 5, the heuristic algorithm is described and the performance is evaluated based on simulation in Section 6. Finally, the conclusion is given in Section 7.

2. Related Work

To deal with the user mobility and variability of SFC requests, there are mainly two thoughts on SFC orchestration optimization. **The first idea attempts to predict the SFC requests arriving model, the volume of data flow and the remaining amount of VNF resources. By reserving resources and calculating SFC routing in advance, resource utilization can be better optimized.** For example, in order to handle the traffic fluctuation, a traffic prediction method is proposed in [10], and two VNF placement algorithms are designed to guide the dynamic VNF instance scaling. To optimize the resource allocation, the resource demand of VNFs is predicted based on machine learning model in the paper [11]. *The performance of prediction based deployment mechanism is closely related to the correctness of prediction methods and the accuracy of historical data sets. At the same time, the diversity of input data will also affect the calculation speed of the algorithms*. Therefore, there are also some studies to optimize SFC deployment as a whole by using reinforcement learning method. In [12], a deep reinforcement learning based algorithm for SFC dynamic orchestration problem with the actor-critic and the deterministic policy gradient scheme is provided, which can efficiently deal with the SFC dynamic deployment in IoT network. In [13] deep reinforcement learning based scheme is proposed as an efficient solution to handle SFC orchestration in dynamic IoT

environment. Experience replay and target network of Deep Q-learning are used to improve the convergence performance of the proposed scheme. *The problem of using reinforcement learning to optimize SFC deployment decision is that the training set is difficult to obtain and the training time of the model is too long. Although a variety of heuristic methods are proposed to speed up model training and improve decision accuracy, due to the particularity of MEC environment, its deployment application also has great problems.*

The other research direction is to transform the long-term running stochastic optimization problem into a series of sub-problems over time slots. At each time slot, the sub-problem is optimized and the VNF migration and real-time SFC adjustment method is used to deal with dynamic [28][29]. In the paper [26], for operational cost minimization of the service chain provider over the entire system span, an efficient online algorithm is proposed and a regularization-based approach from online learning literature to convert the offline optimal deployment problem into a sequence of one-slot regularized problems, each to be efficiently solved in one time slot. In the paper [1], an online orchestration framework for cross-edge service function chaining, which aims to maximize the holistic cost efficiency, via jointly optimizing the resource provisioning and traffic routing on-the-fly is proposed. The long-term cost minimization problem is decomposed into a serious of one-slot fractional problem with a regularization technique and then the fractional solution is rounded to a near-optimal integral solution with a randomized dependent scheme that preserves the solution feasibility. In the paper [27], the system running time is divided into different time slot and in each time slot an Integer Linear Programming model under the constraints of bandwidth and resources in formulated. To compensate for high execution time of the ILP model, two important heuristic algorithms are proposed to distribute and recycle resource efficiently. In [14], the SFCs migration/remapping problem in cloud-fog computing environments is modeled as an integer linear program, then a two-step migration algorithm is proposed to reduce the reconfiguration cost. In [15] dynamic VNF mapping and scheduling are jointly investigated to enhance the performance and a two stage online algorithm is proposed. For the newly arrived SFC, the VNFs are mapped and scheduled by greedily minimizing the waiting time of VNFs. If the delay requirement cannot be satisfied, a delay ware rescheduling scheme is triggered, in which selected existing VNFs are remapped and rescheduled. In [6], a proactive caching-chaining scheme is proposed to support mobility. By proactively performs caching and VNF chaining, the network performance is improved and the service will not be interrupted. In [16], a column generation based heuristic algorithm is used to fulfill new SFC and change existing ones. Considering the complexity of network structure, the mobility of users and the limitation of computing resources, the dynamic readjustment mechanism based on vertical or horizons scaling of VNFs are more suitable for MEC networks. However, to sum up the current researches, there are three deficiencies. *Firstly, the cost of flow migration and other adjustment operations is not considered. Secondly, computing time overhead is relative high. Finally, the mathematical model does not reflect the real network situation of MEC.*

Motivated by the excellent prior researches, the dynamic SFC deployment under the MEC is studied in this paper. The optimization aim is to maximize the profit of service provider by admitting as many as requests as possible and minimize the cost of new request deployment and in service request adjustment. Different from previous studies, we pay more attention to the feasibility and convergence of the mechanism in MEC environment.

3. System Model and Problem Statement

Based on the analysis of SFC request orchestration in MEC network, a system model based on layered graph is built. The SFC dynamic orchestration problem in dynamic environment is clearly defined. For the sake of convenience, a glossary of key symbols used in this paper is listed in **Table 1**.

Table 1. Symbols and description

Notation	Description
$G = (V, E)$	An MEC network G with a set V of network nodes and a set of E links. $V = V_{VNF} \cup V_{FWD}$, V_{VNF} is the service nodes set and V_{FWD} is the forwarding nodes set.
CC_v	The computation resource of node v, $v \in V_{VNF}$.
B_e	The bandwidth resource of link e.
R	SFC request set at a time slot. $R = R_1 \cup R_2$. R_1 is newly arrived request set and R_2 is the changed in-service one.
F	The set of VNF types.
r	A SFC request. A request r is denoted by 5-tuple (s_r, d_r, c_r, B_r, γ_r). s_r and d_r is the access and destination node of r. B_r is the bandwidth request of r and γ_r is the profit of r. c_r denotes the VNF sets requested by r.
f_i^r	The i-th VNF in VNF sequence c_r, where $i = 0, 1, 2... \lvert c_r+1 \rvert$, $f_0^r = s_r$, $f_{\lvert c_r+1 \rvert}^r = s_r$.
$\chi_f^{r,i}$	A Boolean parameter. If the i-th VNF of r is the f-type VNF, $\chi_f^{r,i} = 1$, 0 otherwise.
Δf_i^r	The amount of computation resource required by the i-th VNF in request r per unit bandwidth.
$\theta_{v,f}$	A Boolean parameter. If VNF f can be instantiated in v, $\theta_{v,f} = 1$, 0 otherwise.
$cost(f_i^r, v)$	The processing cost for network function f_i^r on node v are assigned to weight of the directed vertical edge in layered graph G^L.
$\varphi_e^{r,i}$	A Boolean variable. If the route of r goes through link e at layer i of G^L, $\varphi_e^{r,i} = 1$, 0 otherwise.
$\alpha_e^{r,i}$	A Boolean variable. If f_i^r is installed on node v, $\alpha_v^{r,i} = 1$, 0 otherwise.
x_r	A Boolean variable. If the SFC r is served successfully $x_r = 1$, 0 otherwise.
$\omega(v)^+, \omega(v)^-$	The set of out-going and in-coming links of node v in G^L is defined as $\omega(v)^+$ and $\omega(v)^-$.
ρ_1, ρ_2	The unit cost of computation and bandwidth resource is defined as ρ_1, ρ_2.
P_r, p	The feasible path set for request r is defined as P_r and a service path from s_r to d_r is defined as p.
$\alpha_{i,v}^p$	For $p \in P_r$, if the i-th VNF of r is deployed in node v, $\alpha_{i,v}^p = 1$, otherwise 0.
δ_e^p	The number of the occurrences of line e in the path p of r is defined as $\delta_e^p \in N$.
y_p^r	A Boolean variable. If request r is forwarded through service path $p \in P_r$, $y_p^r = 1$, 0 otherwise.

3.1 System Model

Network. The substrate network is defined as $G = (V, E)$, where V denotes the nodes set and E denotes substrate link set. $V = V_{VNF} \cup V_{FWD}$ is defined as the union of two kinds of node sets: the service nodes set V_{VNF}, which provides virtualized platforms for running VNF instances and the forwarding nodes set V_{FWD}, which just only supplies the packet forwarding function. For every node $v \in V_{VNF}$, it supplies some computation and storage resources to support the VNF

instance. The computation resource of v is denoted as CC_v and we also suppose that the storage resource of v is sufficient to support the deployment of all types of VNF [18] [23]. For every link $e \in E$, the bandwidth resource of e is denoted as B_e.

SFC request. In order to optimize the SFC deployment online, the running time of the system is divided into a sequence of time slot [T_1, T_2, T_3…]. An SFC request between a pair of nodes is denoted as r. The SFC requests set at a time slot is defined as $R = R_1 \cup R_2$, which includes newly arrived requests R_1 and the changed in-service ones R_2. Each request r is represented as (s_r, d_r, c_r, B_r, γ_r). The s_r and d_r denotes the source node and destination node of r respectively, where s_r, $d_r \in V$. The c_r is a sequence of specific VNFs which are requested by r and $|c_r|$ denotes the number of VNFs in c_r. The B_r is the bandwidth requirement of r. It is assumed that the resource requirement of VNFs belonging to r is linearly related with B_r [16][23]. The benefit of a successfully deployed request is denoted as γ_r.

The i-th VNF in request r is denoted as f_i^r, where i = 0, 1, 2… $|c_r+1|$, $f_0^r = s_r$, $f_{|c_r+1|}^r = d_r$. A Boolean parameter $\chi_f^{r,i}$ is used to describe the i-th VNF type in r. $\chi_f^{r,i} = 1$, if the i-th VNF of r is a f-type VNF, 0 otherwise. For each request r, there is no duplicated VNFs and we have $\chi_f^{r,i} + \chi_f^{r,j} \le 1, \forall r, f$, if $i \ne j$ [16][1].

VNF. The set of VNF types (e.g., firewall, traffic filter, and video cache) supplied by the network is denoted as F. The amount of computation resource required by VNF type f to process unit bandwidth data is denoted as Δf. The amount of (fractional) computation resource for the i-th VNF in request r is denoted as $B_r \cdot \Delta f_i^r$. We suppose that not all types of VNF can be supported by $v \in V_{VNF}$. For any $f \in F$, $v \in V$, if f can be instantiated in v, $\theta_{v,f} = 1$, 0 otherwise.

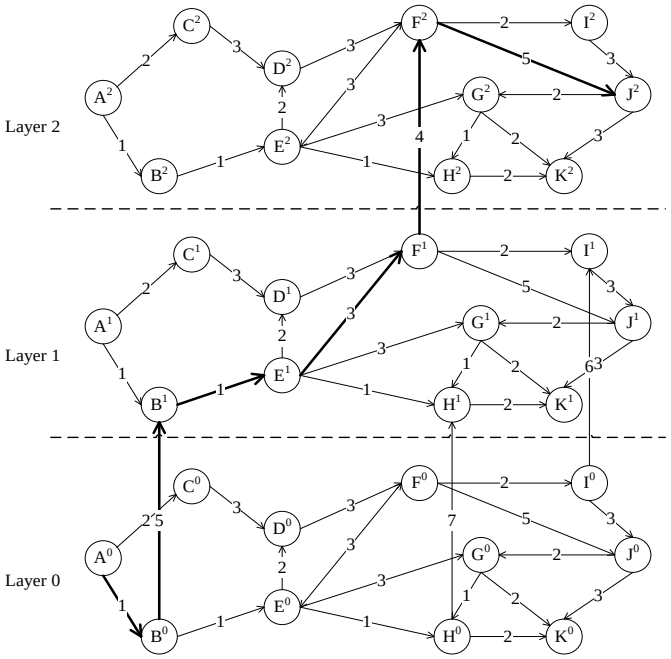


Fig. 2. Layered graph of a SFC request r.

Layered Graph. In order to describe the VNF deployment and flow routing of a specific request r, a layered graph is adopted [17]. The original network G is transformed into a layered graph G^L. The basic graph is treated as layer 0 and other k layers are added into the graph,

where $k = |c_r|$. Each layer is an exact copy of the original graph G. For every vertex v in the original graph, let v_i denotes the corresponding node in the i-th layer. Every (i-1, i) layer pair is connected vertically only by edges between node v^{i-1} and v^i if that nodes provides the particular network function required by f_i^r. A cost function $cost(f_i^r, v)$ which is defined as the processing cost for network function on node v are assigned to weight of the directed vertical edges.

Fig. 2 illustrates how to find a chain placement and routing solution for a request r with service chain $c_r = \{f_1, f_2\}$ and the source node A and the destination node J. A three-layer graph is build based on the original one. The Layer 1 and Layer 2 are the copy of the Layer 0. The function f_1 can be deployed in B, H, I and the function f_2 can be deployed in F. So the Layer 0 is connected to the Layer 1 by $B^0 \rightarrow B^1$, $H^0 \rightarrow H^1$ and $I^0 \rightarrow I^1$. The Layer 1 and the layer 2 are connected by $F^1 \rightarrow F^2$. The edges are weighted by their function processing cost respectively.

3.2 Problem Statement

Based on the system model described above, the **SFC dynamic orchestration in mobile edge computing (SFC-DOMEC)** can be formally described as follows. The system running time is divided into equal interval time slots. At each time slot T, we assume that there is a substrate network $G(V, E)$ and a set of SFC request $R = R_1 \cup R_2$, which includes the newly arrived ones R_1 and the changed in-service ones R_2. For each request r, the SFC-DOMEC problem focuses on chaining VNFs and steering the traffic through these VNFs in order such that the computation resource and bandwidth resource requirements are also satisfied. The objective of SFC-DOMEC is to maximize the profit of service providers under the constraints of computing and bandwidth resources.

4. Optimization Model

In order to maximize the benefit of service provider and minimize the resource cost, the SFC-DOMEC is formulated as an ILP model first, which is called *SFC_ILP* in subsection 4.1 and then to simplify the formulation, a new model based on column generation decomposition model is used to reformulate the problem, called *SFC_CG*, in subsection 4.2.

4.1 Integer Linear Programming Model

Two variable set are used to describe the *SFC_ILP* model. For any request $r \in R$, if the route of r goes through link e at layer i of graph G^L, the Boolean variable $\varphi_e^{r,i}$ =1, 0 otherwise. If f_i^r is installed on node v, $\alpha_v^{r,i}$ =1, 0 otherwise. Note that we will use the convention that if $v \notin V^{VNF}$, $\alpha_v^{r,i} = 0$. A Boolean variables x_r=1 if the SFC r is served successfully, 0 otherwise.

There are four sets of constraints.

1. *VNF Selection Constraint*.

Equation (1) ensures that the required VNFs in c_r must be deployed in the supported node if request r is served at current time slot.

$$\sum_{v \in V_{VNF}} \alpha_v^{r,i} \cdot \theta_{v,f} \cdot \chi_f^{r,i} = x_r, \forall i, \forall r, \forall f \tag{1}$$

2. *Flow Conservation Constraint*.

The flow from source node to destination node must go through the location of the functions of requested in order, which is shown in (2). The set of outgoing links of node v is defined as $\omega(v)^+$ and the in-coming links of node v is defined as $\omega(v)^-$.

$$\sum_{e\in\omega(v)^+}\varphi_e^{r,i}-\sum_{e\in\omega(v)^-}\varphi_e^{r,i}+\alpha_v^{r,i}-\alpha_v^{r,i-1}=0,\quad v\in V,r\in R,0<i<|c_r| \tag{2}$$

Only the source node on the first layer has positive outgoing flow and the destination node on the last layer have a positive incoming flow, which is shown as (3) and (4) respectively.

$$\sum_{e\in\omega(v)^+}\varphi_e^{r,0}-\sum_{e\in\omega(v)^-}\varphi_e^{r,0}+\alpha_v^{r,0}=\begin{cases}1, & if\ v=s_r\\ 0, & otherwise\end{cases},\quad v\in V,r\in R \tag{3}$$

$$\sum_{e\in\omega(v)^+}\varphi_e^{r,|c_r|}-\sum_{e\in\omega(v)^-}\varphi_e^{r,|c_r|}-\alpha_v^{r,|c_r|}=\begin{cases}-1, & if\ v=d_r\\ 0, & otherwise\end{cases},\quad v\in V,r\in R \tag{4}$$

3. *Link capacity constraint*.

For any given link *e*, the sum of bandwidth consumption of its replica at different levels of G^L and cannot exceed the link bandwidth.

$$\sum_{r\in R}B_r\cdot\sum_{i=0}^{|c_r|}\varphi_e^{(r,i)}\le B_e,\quad \forall e\in E \tag{5}$$

4. *Node computation constraint*.

If a function is placed in node *v*, there is a link (v^{i-1}, v^i). The usage of node *v* is described as the weight of (v^{i-1}, v^i) and must less than its capacity.

$$\sum_{r\in R}\sum_{i=0}^{|c_r|}B_r\cdot\Delta f_i^r\cdot\alpha_v^{r,i}\le CC_v,\quad \forall v\in V_{VNF} \tag{6}$$

The aim is to maximize the service provider's profit. In time slot *T*, the set of SFC request is *R*. We have to describe the profit of R_1 and R_2 respectively. The unit cost of computation and bandwidth resource consumption is described by positive coefficients ρ_1 and ρ_2.

1. *Profit of Successfully Deployed SFC Request in* R_1.

The cost of deployment is described as (7).

$$COST_1=\rho_1\cdot\sum_{v\in V_{VNF}}\sum_{r\in R_1}\sum_{i=0}^{|c_r|}B_r\cdot\Delta f_i^r\cdot\alpha_v^{r,i}+\rho_1\cdot\sum_{e\in E}\sum_{r\in R_1}\sum_{i=0}^{|c_r|}B_r\cdot\varphi_e^{r,i} \tag{7}$$

The profit can be described as (8).

$$REV_1=\sum_{r\in R_1}\gamma_r\cdot x_r \tag{8}$$

2. *Profit of Successfully adjusted SFC Request in* R_2.

The deployment cost of R_2 includes two parts. For any request $r\in R_2$, if it cannot be adjusted in time slot *T*, its cost equals to the profit which the service provider received in time slot before and has to pay back for the failure in *T*. This part is described in (9). If the request *r* is deployed successfully, the cost can be described in (10).

$$COST_2^2=\sum_{r\in R_2}(1-x_r)\cdot\gamma_r \tag{9}$$

$$COST_2^1=\rho_1\cdot\sum_{v\in V_{VNF}}\sum_{r\in R_2}\sum_{i=0}^{|c_r|}B_r\cdot\Delta f_i^r\cdot\alpha_v^{r,i}+\rho_2\cdot\sum_{e\in E}\sum_{r\in R_2}\sum_{i=0}^{|c_r|}B_r\cdot\varphi_e^{r,i} \tag{10}$$

The profit of R_2 is 0. If the $r\in R_2$ is deployed successfully, this should be omitted here as its profit has been calculated in the time slot before. If its deployment is failure, the profit is still 0.

By summarizing the analysis above, the finalized optimization objective is described in (11).

$$\textbf{Minimize}\quad COST_1+COST_2^1+COST_2^2-REV_1 \tag{11}$$

Theorem1. *The online* ***SFC-DOMEC*** *problem is NP-hard problem.*

Proof. For the sake of simplicity, the bandwidth constraint is ignored and we set $B_e=+\infty$, $\rho_2=0$. The network supported VNF types and the length of the SFC request are set to be one. In this case, the problem is simplified to a facility location problem with capacity constraints. The

problem is NP hard [18]. As the SFC-DOMEC problem is the restricted case of the general NP-hard problem, it is proved to be NP-hard as well.

4.2 Column Generation Modified Model

The number of variable $\varphi_e^{r,i}$ and $\alpha_v^{r,i}$ becomes very large when the network size of *SFC_ILP* is large. In fact, a feasible solution of the problem is a combination of the feasible deployment scheme of SFC request *R*. An alternate model *SFC_CG* based on column generation is proposed. In column generation model, we use the prim-dual method to obtain the right columns to construct a near optimal solution. The concept of configuration is used in *SFC_CG*, which is a feasible service path for a specific service function sequence.

A service path for request *r* is defined as: (1) a network path, i.e., an ordered set of nodes from the source s_r to the destination d_r of *r*, and (2) a set of node locations for the VNFs in the c_r. Each service path is thus specific to a given request and its SFCs.

We describe the *SFC_CG* as follows.

Parameters. The feasible path set for request *r* is defined as P_r and a service path from s_r to d_r is denoted by $p \in P_r$. The location of the VNFs in service chain is identified by a set of function locations (v, f_i^r). For $p \in P_r$, if the *i*-th VNF of *r* is deployed in node *v*, a Boolean variable $\alpha_{i,v}^p$ is defined to be 1, otherwise 0. The number of the occurrences of line *e* in path *p* is defined as $\delta_e^p \in N$.

Variables. $y_p^r \in \{0,1\}$, where y_p^r =1 if request *r* is forwarded through service path $p \in P_r$, 0 otherwise.

Based on the definition of new variables and parameters, the optimization objective can be rewrite as follows.

The deployment cost and profit of R_1 can be rewritten as (12) and (13).

$$COST_1 = \rho_1 \cdot \sum_{r \in R_1} \sum_{p \in P_r} \sum_{i=0}^{|c_r|} \sum_{v \in V_{VNF}} B_r \cdot \Delta f_i^r \cdot \alpha_{i,v}^p \cdot y_p^r + \rho_2 \cdot \sum_{r \in R_2} \sum_{p \in P_r} \sum_{e \in E} B_r \cdot \delta_e^p \cdot y_p^r \tag{12}$$

$$REV_1 = \sum_{r \in R_1} \sum_{p \in P_r} \gamma_r \cdot y_p^r \tag{13}$$

The deployment cost of R_2 is rewritten as (14) and (15).

$$COST_2^1 = \rho_1 \cdot \sum_{r \in R_2} \sum_{p \in P_r} \sum_{i=0}^{|c_r|} \sum_{v \in V_{VNF}} B_r \cdot \Delta f_i^r \cdot \alpha_{i,v}^p \cdot y_p^r + \rho_2 \cdot \sum_{r \in R_2} \sum_{p \in P_r} \sum_{e \in E} B_r \cdot \delta_e^p \cdot y_p^r \tag{14}$$

$$COST_2^2 = \sum_{r \in R_2} \gamma_r \cdot \sum_{p \in P_r} (1 - y_p^r) \tag{15}$$

Objective. The SFC_CG objective function is described as (16).

$$\begin{aligned} \textbf{Minimize} \quad & \sum_{r \in R} \sum_{p \in P_r} y_p^r \cdot \sum_{i=1}^{|c_r|} \sum_{v \in V_{VNF}} \rho_1 \cdot B_r \cdot \Delta f_i^r \cdot \alpha_{i,v}^p + \sum_{r \in R} \sum_{p \in P_r} y_p^r \cdot \sum_{e \in E} \rho_2 \cdot B_r \cdot \delta_e^p \\ & - \sum_{r \in R} \sum_{p \in P_r} y_r^p \cdot \gamma_r + \sum_{r \in R_2} \gamma_r \end{aligned} \tag{16}$$

Removing the constant term, the objective function can be further simplified as (17).

$$\textbf{Minimize} \quad \sum_{r \in R} \sum_{p \in P_r} \left(\sum_{v \in V_{VNF}} \left(\sum_{i=1}^{|c_r|} \rho_1 \cdot B_r \cdot \Delta f_i^p \cdot \alpha_{i,v}^p \right) + \sum_{e \in E} \rho_2 \cdot B_r \cdot \delta_e^p - \gamma_r \right) \cdot y_p^r \tag{17}$$

Constraints.

1. Exactly one path per request.

$$0 \le \sum_{p \in P_r} y_p^r \le 1 \tag{18}$$

2. Link capacity constraint.

$$\sum_{r\in R}\sum_{p\in P_r}(B_r\cdot\delta_e^p)\cdot y_p^r \le B_e, \forall e \tag{19}$$

3. Node capacity constraint.

$$\sum_{r\in R}\sum_{i=0}^{|c_r|}\sum_{p\in P_r}(B_r\cdot\Delta f_i^r\cdot\alpha_{i,v}^p)\cdot y_p^r \le CC_v, \forall v\in V_{VNF} \tag{20}$$

5. Column Generation Based Heuristic Algorithm

As described in sub-section 4.2, each candidate path is treated as a column and it can be easily solved by column generation algorithm [18] [19]. The CG algorithm is a deformation of simplex method. Based on a primal-dual decomposition technique, the Linear Master Problem (*LMP*) is divided into a Restricted Linear Master Problem (*RLMP*), which selects the best configurations and an Integer Linear Programming Pricing Problems (*ILP_PP*), which generates feasible configuration for each service. We solve the *RLMP* and *ILP_PP* iteratively until there is no any improving configuration for *ILP_PP* anymore. Finally, we use an ILP solver on the last *RLMP* to derive a linear relaxation solution of the model SFC_CG.

Algorithm 1. *SFC_DOMEC_Solver*

Input: $G(V, E)$, R, CC_v, B_e, F, s_r, d_r, c_r, B_r, γ_r, Δf, $\theta_{v,f}$, $\chi_f^{r,i}$, ρ_1, ρ_2

Output: Feasible solution of *SFC_CG*

1. Define the notes of a column and formulate the *ILP* model *SFC_CG* for *MP*.
2. Relaxing the integer variable to get *LMP*.
3. Generate an initial set of columns Ω which satisfy the constraints and change the *LPM* to *RLMP*.
4. **while** *true* **do**
5. Solve the *RLPM* to get the values of primal and dual variables.
6. Construct sub problems *ILP_PP* with the dual variables and get its optimization objective ¤ .
7. **if** ¤ ≥ 0 **then**
8. **break**
9. **end if**
10. Generate new columns with the result of *ILP_PP* and insert the columns to Ω.
11. Use Ω to rebuild RLMP again.
12. **end while**
13. Use Ω to transform *RLPM* to *SFC_CG* and solve it.

Based on the principle of CG algorithm and SFC_CG model, the solution of SFC_DOMEC is shown in Algorithm 1. The time complexity of the column generation is analyzed in [18].

5.1 Pricing Problem

A feasible service path for a request can be derived by the pricing problem and the objective of pricing problem is the reduced cost of the primal linear programming. We defined the dual variables u_r, u_e and u_v for the constraints (18), (19) and (20).

Variables. If f_i^r is deployed on node v, we defined $\alpha_v^i=1$, 0 otherwise. $\varphi_e^i\in\{0,1\}$, where $\varphi_e^i=1$ if the flow forwarded on link e on layer i of the layered graph G^L, 0 otherwise.

The service path generator for each request r model is described as (21).

$$\textbf{Minimize} \sum_{e\in E}\sum_{i=0}^{|c_r|} B_r\cdot(\rho_2+u_e)\cdot\varphi_e^i+\sum_{v\in V_{VNF}}\sum_{i=0}^{|c_r|} B_r\cdot\Delta f_i^r\cdot(\rho_1+u_v)\cdot\alpha_v^i+u_r-\gamma_r \tag{21}$$

Constraint.

Flow conservation. The flow from function i to function i+1 of the SFC r must satisfy constraints (22) and then the flow from the source node to the first VNF deployed node must satisfy constraint (23).

$$\sum_{e\in\omega(v)^+}\varphi_e^i-\sum_{e\in\omega(v)^-}\varphi_e^i+\alpha_v^i-\alpha_v^{i-1}=0,\quad v\in V,0<i\lhd|c_r| \tag{22}$$

$$\sum_{e\in\omega(v)^+}\varphi_e^0-\sum_{e\in\omega(v)^-}\varphi_e^0+\alpha_v^0=\begin{cases}1, & if\ v=s_r\\ 0, & otherwise\end{cases} \tag{23}$$

Similarly, the constraint (24) constrains the flow from the location of the last VNF to the destination d_r.

$$\sum_{e\in\omega(v)^+}\varphi_e^{|c_r|}-\sum_{e\in\omega(v)^-}\varphi_e^{|c_r|}-\alpha_v^{|c_r|}=\begin{cases}-1, & if\ v=d_r\\ 0, & otherwise\end{cases} \tag{24}$$

Link capacity. For $e \in E$, we have (25).

$$B_r\cdot\sum_{i=0}^{|c_r|}\varphi_e^i\le B_e,\quad \forall e\in E \tag{25}$$

Node capacity. For $v \in V_{VNF}$, we have (26).

$$\sum_{i=0}^{|c_r|}(B_r\cdot\Delta f_i^r)\cdot\alpha_v^i\le CC_v,\quad \forall v\in V_{VNF} \tag{26}$$

The i-th VNF of r must be instantiated on nodes which support this type of VNF (27).

$$\sum_{v\in V_{VNF}}\alpha_v^i\cdot\theta_{v,f}\cdot\chi_f^{r,i}=1 \tag{27}$$

5.2 Greedy Based Solution

In Algorithm 1, there are two key problems need to solved. The first is how to find the initial feasible configurations (also defined as the columns of the constraints matrix). The second is how to speed up the solutions of *ILP_PP* problem. By analyzing the objective function of LMP and ILP_PP, it can be seen that to deploy the requests with high profit/cost as early as possible can better and quicker approach the optimal goal. Based on this inspection, a heuristic algorithm based on greedy idea is designed.

1. Initial Configuration Generation Algorithm.

To calculate the initial of feasible configuration set, we design a heuristic algorithm based on greedy idea as shown in Algorithm 2. For $\forall r,r'\in R$, i =1, 2, …$|c_r|$, if the VNF type of c_i^r is the same as $c_i^{r'}$ and $|c_r|=|c_{r'}|$, we say that they belong to the same ***SFC request group***, which is denoted as π_r or $\pi_{r'}$.We use Π_r to denote all the SFC groups of R.

Algorithm 2. *Initial Configuration Generation Algorithm*

Input: $G(V, E)$, R, CC_v, B_e, ρ_1, ρ_2
Output: Ω : Initial set of columns for *RLPM*
1. Generate SFC group set Π of R and $\Omega\leftarrow\varnothing$
2. $\Gamma\leftarrow\varnothing$
3. **for** $\forall$ group $\pi\subset\Pi$ **do**
4. $\pi_{\min}\leftarrow\infty$
5. **for** $\forall$ request $r\in\pi$ **do**

6. **if** $B_r / \gamma_r < \pi_{\min}$ **then**
7. $\pi_{\min} \leftarrow B_r / \gamma_r$
8. **end if**
9. **end for**
10. Add the request *r* of π with the minimal $\pi_{\min}$ to Γ
11. **end for**
12. $r \leftarrow$ the request *r* with minimal B_r / γ_r in Γ
13. **while** $\Gamma \neq \varnothing$ **do**
14. Generate the layered graph G^L of request *r*
15. **for** $\forall e \in E$ **do**
16. set $cost_e^i \leftarrow \infty$, which is the cost of the link corresponding to *e* in the *i*-th layer of G^L
17. **if** $B_e - B_r > 0$ **then**
18. $cost_e^i \leftarrow \rho_2$
19. **end if**
20. **end for**
21. **for** $\forall v \in V$ **do**
22. $cost(f_i^r, v) \leftarrow \infty$
23. **if** $CC_v - B_r \cdot \sum_{i=0}^{|c_r|} \Delta f_i^r \cdot \theta_{v, f_i^v} > 0$ **then**
24. $cost(f_i^r, v) \leftarrow \rho_1$
25. **end if**
26. **end for**
27. Calculate the shortest path $path_r$ from s_r to d_r
28. add the columns corresponding to $path_r$ to Ω
29. **for** $\forall e \in E$ **do**
30. $B_e \leftarrow B_e - B_r \cdot \sum_{i=0}^{|c_r|} \varphi_e^{r,i}$
31. **end for**
32. **for** $\forall v \in V$ **do**
33. $CC_v \leftarrow CC_v - \sum_{i=0}^{|c_r|} B_r \cdot \Delta f_i^r \cdot \alpha_v^i$
34. **end for**
35. $r \leftarrow$ the request *r* with minimal B_r / γ_r in $\Gamma \setminus r$
36. **end while**
37. **return** Ω

In Algorithm 2, the SFC group set Π is generate for the request set *R* for some time slot and set the initial column set Ω for RLPM to be $\varnothing$. We use Γ to denote the request set, which includes the request with the minimal B_r / γ_r of each group and initiate Γ to be $\varnothing$ (*line* 1 to *line* 2). For each group $\pi \subset \Pi$, we find the request $r \in \pi$, which has the minimal bandwidth and maximum profit (*line* 3 to *line* 10). The requests in Γ are sorted in ascending order according to the value of B_r / γ_r and perform the following operations for each request *r* in turn. Generate the layered graph G^L for request *r* (*line* 14). For $\forall e \in E$, firstly, we set its weight to be $\varnothing$, then we check if $B_e - B_r > 0$, which means if the remaining bandwidth of *e* satisfies the bandwidth requirement of *r*. If so, we set the weight of $cost_e^i$ in each *i* of the layered graph to be ρ_2 (*line* 15

to *line* 20). For $\forall v \in V$, we set its weight to be $\varnothing$, then we check if $CC_v - B_r \cdot \sum_{i=0}^{|c_r|} \Delta f_i^r \cdot \theta_{v,f_i^v} > 0$, which means if the remaining computation resource of v satisfies the computation resource requirement of r. If so, we set the weight *cost* (f_i^r, v) which is defined in subsection 3.1 to be ρ_1 (*line* 21 to *line* 26). Based on this weighted layered graph, we calculate the shortest path $path_r$ from s_r to d_r based on the Dijkstra algorithm [21]. The columns corresponding to the $path_r$ is added to Ω (*line* 27 to *line* 28). The remaining bandwidth of the link e is updated $B_e \leftarrow B_e - B_r \cdot \sum_{i=0}^{|c_r|} \varphi_e^{r,i}$ (*line* 29 to *line* 30) and the remaining computation resource of node v is updated by $CC_v \leftarrow CC_v - \sum_{i=0}^{|c_r|} B_r \cdot \Delta f_i^r \cdot \alpha_v^i$ (*line* 32 to *line* 34). Then the next request is processed. If Algorithm 2 still cannot find a feasible solution, it will degenerate to the exhaustive algorithm used in [20]. The time complexity of the Algorithm 2 comes from the calculation from line 13 to 36. For the request r with minimal B_r / γ_r, the Dijkstra algorithm is used to calculate the shortest path. The longest request chain length is denoted as K. When the heap binary data structure is used, the time complexity of Dijkstra in the layer graph G^L is O ($|KV|$ log $|KV|+|KE|$). The maximum run time of Algorithm 2 is O ($|T|$ ($|KV|$ log $|KV|+|KE|$)).

2. Heuristic Algorithm for Accelerating *PP*.

At the same time, in order to further speed up the solution of *PP* problem, we analyze the optimization objective *PP*, and we can see that processing the SFC request with high benefit/cost ratio first can greatly reduce the number of loops and approaching the objective faster. Since the solution for the pricing problems of each request is independent, we solve the corresponding ILPs in parallel to reduce the overall computation time.

A constrained shortest path in layered graph is used to solve the problem more quickly. The weight ω_{ie} of the link e at layer i of the layered graph of the user r is equal to (28) according to (21).

$$\omega_{ie} = B_r \cdot (\rho_2 + u_e) \tag{28}$$

The weight of the cross-layer link (v^i, v^{i+1}) is equal to (29) according to (21).

$$\omega_{iv} = B_r \cdot \Delta f_i^r \cdot (\rho_1 + u_v) \tag{29}$$

The heuristic algorithm for accelerating the *PP* problem is shown in Algorithm 3.

Algorithm 3. *Heuristic Algorithm to Solver ILP_PP*

Input: $G(V, E)$, R, CC_v, B_e, ρ_1, ρ_2, Ω, dual variables u_r, u_e, u_v
Output: Ω : set of columns for *RLPM*

1. $r \leftarrow$ the request r with minimal B_r / γ_r in R
2. Construct *ILP_PP*(r) problem with the dual variables from *RLPM* (21)-(27)
3. **While** $R \neq \varnothing$ **do**
4. Construct the layered graph G^L of r
5. Set the weight ω_{ie} of the link e at layer i of G^L to be $B_r \cdot (\rho_2 + u_e)$ according to (28) and set the weight of cross-layer link (v^i, v^{i+1}) at node v to be $B_r \cdot \Delta f_i^r \cdot (\rho_1 + u_v)$ according to (29)
6. Calculating the weighted shortest $path_r$ from s_r to d_r of r in the layered graph G^L
7. Calculating the objective value Q_r of r according to (21)
8. **if** Q_r <0 **then**
9. Translate the $path_r$ to a column and add the column to Ω'
10. **end if**
11. $r \leftarrow$ the request r with minimal B_r / γ_r in $R \setminus r$
12. **end while**
13. **if** $\Omega' = \varnothing$ **then**

14. **break**
15. **end if**
16. **return** $\Omega \leftarrow \Omega' + \Omega$

The time complexity of the Algorithm 3 is analyzed like this. For each request, the weighted shortest path is calculated by Dijkstra algorithm in the layered graph G^L, so the maximum run time of Algorithm 3 is O ($|R|$ ($|KV|$ log $|KV|+|KE|$)).

6. Performance Evaluation and Discussion

In this section, the performance of proposed algorithm is evaluated based on numerical simulations. The comparison algorithms are shown in subsection 6.1. Four performance metrics are defined in subsection 6.2 followed by a description of the experimental setup in subsection 6.3. Finally, we present the result found through extensive simulation in subsection 6.4.

6.1 Comparison Algorithms

Three existing algorithms are used to compare with proposed algorithm.

Greedy based algorithm (*Greedy*). The algorithm sorts all requests in ascending order according to their traffic volume, and constructs a layered graph for each request in turn. For each VNF, the node with the most remaining resources is selected for mapping. Considering the residual bandwidth of links and bandwidth requirement of SFC, the greedy algorithm uses a modified Dijkstra algorithm to calculate path between two adjacent selected service nodes.

SFC_ILP algorithm (*SFC_ILP*). We use the interactive optimizer CPLEX to solve ILP model, and use the result to deployment each SFC in each time slot.

Viterbi based algorithm (*Viterbi*). With real-time perception of the resource change of underlying node, the topology structure will be adjusted dynamically. Hidden Markov model is used to describe the topology in formation of available node with resource constraints in underlying network. Based on multi-state graph, the provisioning cost for each request is calculated and sorted in descending order. Then Viterbi is used to serves them one by one [22].

6.2 Performance Metrics

In this paper, we defined 4 kinds of metrics to evaluate the performance of the algorithm.

Service Mapping Time. It defines the mapping time of a new SFC.

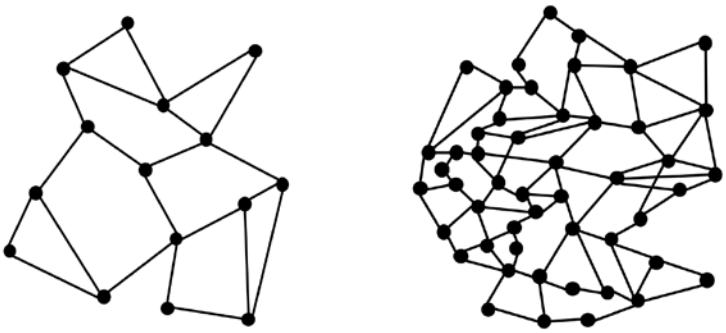
Service Provider's profit. It refers to the profit of all requests which can be deployed minus the resource cost.

Success Ratio of Mapping. It is defined as (# of successful mapped SFCs)/ (# of all request).

Success Ratio of Redeploying. It is defined as (# of successful remapped SFCs)/ (# of all changed SFCs).

6.3 Evaluation Setup

The simulation is implemented in Matlab2020a with the CPLEX toolbox to solve the ILP and CG models, and conducted in a computer with Inter(R) Xeon CPU W5580 @3.2GHZ and 64 GB of RAM.



(a) Atlanta network. (b) Germany50 network.

Fig. 3. Network used in simulation

Network Topology. The simulations are conducted on an Atlanta topology and the Germany50 network which are shown in **Fig. 3** [23]. In the topology, the nodes with higher node degree are selected as service node. All nodes are sorted in descending order based on node degree, and then select the first 5 nodes and 20 nodes as service nodes for Atlanta and Germany50, respectively. Through the normalization method [24], we replace specific metrics with 'unit'. The IT resource of service nodes are randomly numbers uniformly distributed between 1000 and 1500 unit. The bandwidth per link is 1000 units. The cost of ρ_1 and ρ_2 is set to be the same value 0.1.

VNF set. There are 10 types of VNFs which are supported in the network. The amount of computation resource per unit bandwidth required by each VNF is set to randomly distributed between (3, 5). We assume that the types of VNFs that can be deployed in each service node are uniformly distributed between (4, 5). The maximum number of VNF instances permitted to place per service node is set as 10 [25].

SFC request data set. To simulate the user request, the self-similar traffic model in [15] is leveraged. The online service request arrives following a Poisson process with an average rate of 5 SFC in every time units and the service time follows an exponential distribution with an average of 200 time units. The bandwidth required of each SFC is a number with uniform distribution between (10, 20) units. The benefit of each successful deployed request is a number with uniform distribution between (30,100). The source and destination of each SFC were randomly selected in the simulation experiment. The VNF number in each SFC was randomly distributed in (3, 5), and VNF type is different.

6.4 Performance

6.4.1 Static Operation Comparison

To evaluate the static performance of the proposed algorithm, we first evaluate the SFC provisioning at one time slot. Here, we assume that SFC requests $R = R_1$, that is to say, for an empty network environment without any requests deployed in advance, we will evaluate the change of the designed algorithm in computing time and provider's actual benefit. The evaluation is carried out by varying the number of requests and the upper limit of the computation time is set to be 200.

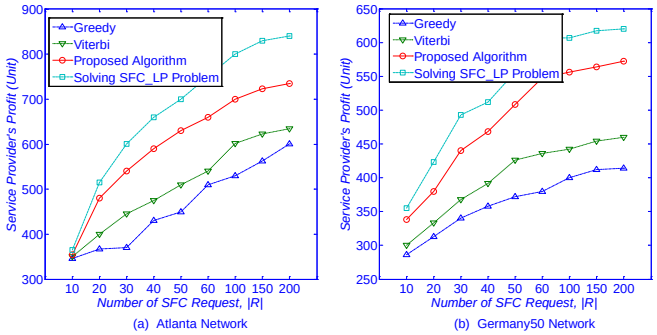


Fig. 4. Service provider's total profit in one-time operation

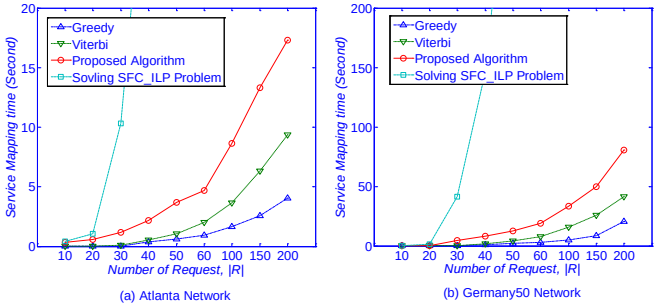


Fig. 5. Total deployment time in one-time operation

Fig. 4 shows the comparison of service provider profit under different topology and different number of user request. In **Fig. 4**, we also add the calculation result of the linear programming model *SFC_LP* after the relaxation of model *SFC_ILP*. We observe that the *SFC_LP* solution always lays above all other algorithms. This is because the LP relaxation solution provides an upper bound on the service provider's profit but might not be a feasible solution to the original problem. Hence, we can use it as a baseline to evaluate the algorithms' performance the LIP is intractable. It show that when the number of user requests is less than 10, the performance of the algorithms is similar, but when SFC number increases, the advantages of the proposed algorithm in the service provider profit are obvious. This is because Viterbi and greedy based algorithms are heuristic algorithms. When the scale of the problem becomes larger, they are easy to fall into location optimum, so they cannot follow the growth trend of ILP as CG algorithm does. It can be seen that the algorithm's superior performance in service provider's profit is actually at the cost of increasing time complexity. We have carried out experimental simulation on Atlanta and Germany50 respectively. The results are shown in **Fig. 4** (a) and **Fig. 4** (b). The trend of change between different algorithms is similar in two types of topology, which shows that the proposed algorithm in this paper will certainly perform well in a network with practical topology structure and larger network scale. However, we can also see that as the network topology becomes more complex, the maximum revenue that the service provider can obtain gradually becomes smaller. This is mainly because with the increase of network topology, the path between different service functions becomes longer, which makes the network bandwidth overhead larger and leads to the decline of service providers' profits. We find that the performance of the proposed algorithm is at least 30% better than Viterbi algorithm under two kinds of topology and different user requests.

Fig. 5 (a) and **Fig. 5** (b) compare the time cost of the proposed algorithm and the comparison algorithm under the topology of Atlanta and German50 with different number of request. The computation time of *SFC_ILP* is comparable to CG when the number of request is less than 22. This is because *SFC_ILP* computation time is related with $|E|\times|R|\times|c_r|\times|F|$ [18]. When the request number is small, the *CG* computation is approximate the solution of *ILP* with iterations. In the Germany 50 topology, when the number of requests exceeds 30, the computing time has exceeded the limit of 200s. This is mainly because the topology of Gernamy50 is large, which need more time to converge. Since Viterbi and greedy algorithm do not use iterative method, their running speed is obviously faster than the algorithm proposed in this paper, which confirms our analysis that column generation can obtain higher profits for service providers at the cost of increasing time complexity. Therefore, we should give priority to Viterbi algorithm when the performance gap between the algorithm proposed in this paper and Viterbi algorithm is small in the profit of service providers. For the actual network operation, the delay is intolerable. In real network environment, it can be improved by two methods. Firstly, it can be improved by applying more efficient programming way. Secondly, if we improve the performance of computing platform, such as cloud computing platform, we can further reduce the computing time, which is easy for Internet Service Providers.

6.4.2 Dynamic Operation Comparison

Based on the static performance analysis of the algorithm, we also test the performance of the algorithm in the dynamic running environment. We assume that at the beginning of each service time slot, the service provider calculates a new deployment scheme for all new arrival service requests and requests on-the-fly which have changed. In order to process new requests timely and reduce service waiting delay, we set the service time slot as 3 time units. The arrival rate of service requests in the unit time slot is set to be Poisson distribution with mean value of 30, and the average operation time of each service is subject to the exponential distribution of 15 time units. We set that 1/3 of the services in each time slot change. Based on the parameter setting of the simulation experiment, the running data of 1500 time units are generated. Based on this data, the comparison of the proposed algorithm with the Viterbi and Greedy based algorithm on the metrics of *Success Ratio of Mapping* and *Success Ratio of Redeploying* are compared on the topology of Atlanta and Germany50 respectively.

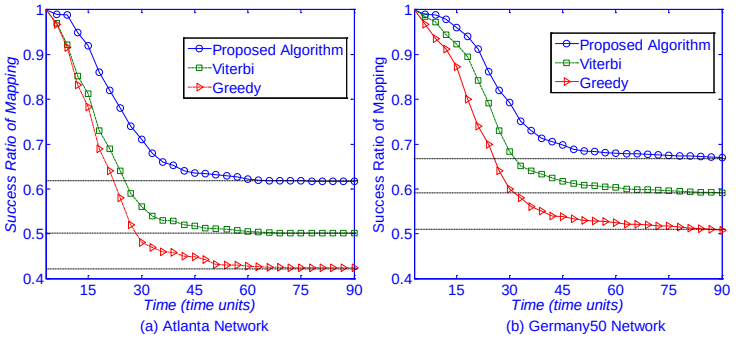


Fig. 6. Success ratio of mapping in Atlanta and German50 networks

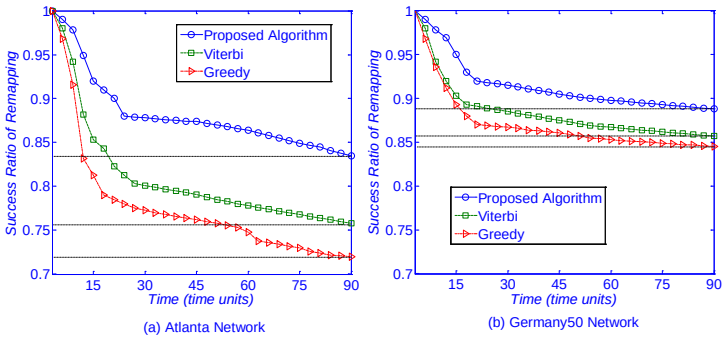


Fig. 7. Success ratio of redeploying in Atlanta and Germany50 Networks.

The success ratio of mapping and success ratio of redeploying is shown in **Fig. 6** and **Fig. 7**. As shown in **Fig. 6**, the trend of the success ratios of mapping of all algorithms is decreasing. This is because with the increasing of the deployed SFC number, the available network resource is reducing. The success ratio of mapping of proposed algorithm drops less, which indicates that proposed algorithm performs well in the network with heavy load. Additionally, the success ratio of mapping sometimes may increase. This is because more resource will release as some SFCs terminate or the bandwidth requirements of some SFCs decreasing.

As shown in **Fig. 7**, the changing trend of success ratio of redeploying of all algorithms is similar to that of the mapping success rate performance in **Fig. 6**. As can be seen from **Fig. 6**, compared with Viterbi algorithm, the average deployment success rate of the proposed algorithm is about 12% higher than that of Viterbi algorithm. As can be seen from **Fig. 7**, compared with Viterbi algorithm, the algorithm proposed in this paper is about 10% higher in the re deployment of variable SFC.

7. Conclusion

In this paper, we consider the orchestration problem of multiple SFC requests in the mobile edge computing environment from the perspective of improving profit and reducing the cost of service providers. An integer linear programming model is constructed based on layered graph, and a new optimization model is proposed by using the idea of column generation algorithm, and the model is improved by using the idea of column generation algorithm. Based on the column generation model, a new SFC orchestration mechanism based on greedy algorithm and column generation algorithm is proposed. The numerical simulation results show that our algorithm is faster than directly solving the ILP model in the case of closer to the optimal solution, and can meet the actual network operation requirements. The main disadvantage of this paper is that we did not consider the VNF migration cost and service delay constraints in our experiments. Therefore, we need to further study the dynamic SFC orchestration mechanism by considering the cost of SFC migration when the SFC request delay is constrained, so as to further improve the model.

References

[1] Z. Zhou, Q. Wu and X. Chen, “Online Orchestration of Cross-Edge Service Function Chain for Cost-Efficient Edge Computing,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no.8, pp.1866-1880, Jul. 2019. Article (CrossRef Link).

[2] J. Li, W. Liang and Y. Ma, "Robust Service Provisioning With Service Function Chain Requirements in Mobile Edge Computing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 2138-2153, Jun. 2021, Article (CrossRef Link).
[3] Y. Chen and W. Liao, "Mobility-aware Service Function Chaining in 5G Wireless Networks with Mobile Edge Computing," in *Proc. of 2019 IEEE International Conference on Communications (ICC), Shanghai*, Pudong, China, pp. 1-6, 2019.
[4] X. Wei, J. Liu, Y. Wang, C. Tang and Y. Hu, "Wireless edge caching based on content similarity in dynamic environments," *Journal of Systems Architecture*, vol.115, pp.1-8, May. 2021. Article (CrossRef Link).
[5] A. M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, and T. Magedanz, "Service Function Chaining in Next Generation Networks: State of the Art and Research Challenges," *IEEE Communications Magazine*, vol. 55, no.2, pp. 216-223, Oct. 2016. Article (CrossRef Link).
[6] D. T. Nguyen, C. Pham and K. K. Nguyen et al., "Placement and Chaining for Run-time IoT Service Deployment in Edge-Cloud," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 459-472, Mar. 2020. Article (CrossRef Link)
[7] G. Zheng, A. Tsiopoulos and V. Friderikos, "Dynamic VNF Chains Placement for Mobile IoT Applications," in *Proc. of 2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1-6, 2019. Article (CrossRef Link).
[8] Y. Liu, H. Lu, X. Li, D. Zhao, W. Wu and G. Lu, "A Novel Approach for Service Function Chain Dynamic Orchestration in Edge Clouds," *IEEE Communications Letters*, vol. 24, no. 10, pp. 2231-2235, Oct. 2020, Article (CrossRef Link).
[9] A. Jarray and A. Karmouch, "Decomposition Approaches for Virtual Network Embedding With One-Shot Node and Link Mapping," *IEEE/ACM Transactions on Networking*, vol. 23, no. 3, pp. 1012-1025, Jun. 2015, Article (CrossRef Link).
[10] H. Tang, D. Zhou and D. Chen, "Dynamic Network Function Instance Scaling Based on Traffic Forecasting and VNF Placement in Operator Data Centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 3, pp. 530-543, 1 Mar. 2019. Article (CrossRef Link).
[11] H. Kim, S. Jeong, D. Lee, H. Choi, J. Yoo and J. W. Hong, "A Deep Learning Approach to VNF Resource Prediction using Correlation between VNFs," in *Proc. of 2019 IEEE Conference on Network Softwarization (NetSoft)*, pp. 444-449, 2019. Article (CrossRef Link).
[12] Y. Liu, H. Lu, X. Li, Y. Zhang, L. Xi and D. Zhao, "Dynamic Service Function Chain Orchestration for NFV/MEC-Enabled IoT Networks: A Deep Reinforcement Learning Approach," *IEEE Internet of Things Journal*, vol. 8, no. 9, pp. 7450-7465, 1 May. 2021. Article (CrossRef Link).
[13] X. Fu, F. R. Yu, J. Wang, Q. Qi and J. Liao, "Dynamic Service Function Chain Embedding for NFV-Enabled IoT: A Deep Reinforcement Learning Approach," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 507-519, Jan. 2020. Article (CrossRef Link).
[14] D. Zhao, G. Sun, D. Liao, S. Xu, and V. Chang, "Mobile-aware service function chain migration in cloud-fog computing," *Future Generation Computer Systems*, vol.96, pp.591-604, Jul.2019. Article (CrossRef Link).
[15] J. Li, W. Shi, P. Yang and X. Shen, "On Dynamic Mapping and Scheduling of Service Function Chains in SDN/NFV-Enabled Networks," in *Proc. of 2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1-6, 2019. Article (CrossRef Link).
[16] J. Liu, W. Lu, F. Zhou, P. Lu and Z. Zhu, "On Dynamic Service Function Chain Deployment and Readjustment," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 543-553, Sept. 2017. Article (CrossRef Link).
[17] A. Dwaraki and T. Wolf, "Adaptive Service-Chain Routing for Virtual Network Functions in Software-Defined Networks," in *Proc. of the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization(HotMiddlebox '16)*, Florianopolis Brazil, pp. 32-37, Aug. 2016. Article (CrossRef Link).
[18] N. Huin, B. Jaumard and F. Giroire, "Optimal Network Service Chain Provisioning," *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1320-1333, Jun. 2018. Article (CrossRef Link).

[19] H. B. Amor, J. Desrosiers and J. M. Carvalho, "Dual-Optimal Inequalities for Stabilized Column Generation," *Operations Research, INFORMS*, vol. 54(3), pp. 454-463, June 2006. Article (CrossRef Link).
[20] H. A. Alameddine, S. Sebbah and C. Assi, "On the Interplay Between Network Function Mapping and Scheduling in VNF-Based Networks: A Column Generation Approach," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 860-874, Dec. 2017. Article (CrossRef Link)
[21] M. Barbehenn, "A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices," *IEEE Transactions on Computers*, vol. 47, no. 2, pp. 263, Feb. 1998. Article (CrossRef Link).
[22] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba and O. C. M. B. Duarte, "Orchestrating Virtualized Network Functions," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 725-739, Dec. 2016. Article (CrossRef Link).
[23] T. Sato, A. Kikuchi, R. Shinkuma and E. Oki, "Column Generation Based Algorithm for Service Chaining Relaxing Visit Order and Routing Constraints," in *Proc. of IEEE Global Communications Conference*, pp. 1-6, 2020. Article (CrossRef Link).
[24] L. Wang, Z. Lu, X. Wen, R. Knopp and R. Gupta, "Joint Optimization of Service Function Chaining and Resource Allocation in Network Function Virtualization," *IEEE Access*, vol. 4, pp. 8084-8094, Nov. 2016. Article (CrossRef Link).
[25] S. Mehraghdam, M. Keller and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. of 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, pp. 7-13, 2014. Article (CrossRef Link).
[26] Y. Jia, C. Wu, Z. Li, F. Le and A. Liu, "On line Scaling of NFV Service Chains Across Geo-Distributed Datacenters," *IEEE/ACM Trans. Netw*, vol.26, no.2, pp. 699-710, Arp.2018.
[27] Y. Liu, H. Zhang, H. Guan and Y. Wang, "A Method for Adaptive Resource Adjustment of Dynamic Service Function Chain," *IEEE Access*, vol. 6, pp. 69988-70004, 2018. Article(CrossRef Link).
[28] X. Huang, S. Bian, X. Gao, W. Wu, Z. Shao and Y. Yang, "Online VNF Chaining and Scheduling with Prediction: Optimality and Trade-Offs," in *Proc. of 2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1-6, 2019. Article(CrossRef Link).
[29] Z. Luo and C. Wu, "An Online Algorithm for VNF Service Chain Scaling in Datacenters," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1061-1073, Jun. 2020. Article(CrossRef Link).
[30] T. Nguyen, E. Huh and M. Jo, "Decentralized and Revised Content-Centric Networking-Based Service Deployment and Discovery Platform in Mobile Edge Computing for IoT Devices," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4162-4175, Jun. 2019. Article(CrossRef Link).

Xiulei Wang received the B.S. degree from College of Computer Science and Technology, Sichuan University in 2009 and Ph.D. degrees from College of Command Information System, PLA University of Science and Technology in 2016. He is currently an Associate Professor in Institute of Command and Control Engineering, Army Engineering University. His current research interests include network function virtualization, mobile edge computing and network management.
Email: xiuleiwang1988@126.com.

Bo Xu received the B.S. and Ph.D. degrees from College of Command Information System, PLA University of Science and Technology in 2006 and 2011, respectively. He is currently an Associate Professor in Institute of Command and Control Engineering, Army Engineering University. His current research interests include network twin technology, network performance measurement and network management.
Email: xubo820@163.com.

Fenglin Jin received the B.S. degree from institute of Command Automation, PLA University of Science and Technology in 2001 and Ph.D. degree from Computer Science of Nanjing University in 2013. He is currently a Professor in Institute of Command and Control Engineering, Army Engineering University. His current research interests include computer network, space earth integration network and satellite communications.
Email: 18951003070@189.cn.