Speedy and Efficient Service Chaining and Function Placement Based on Lagrangian Heuristics for Capacitated Shortest Path Tour Problem [‡]

Takanori Hara[1*†] and Masahiro Sasabe[1†]

[1*]Division of Information Science, Nara Institute of Science and Technology, 8916-5 Takayama-cho, Ikoma, 630-0192, Nara, Japan.

*Corresponding author(s). E-mail(s): hara@ieee.org;
Contributing authors: m-sasabe@ieee.org;
†These authors contributed equally to this work.

Abstract

Network functions virtualization (NFV) can realize flexible and diverse network services by replacing the conventional network equipment with the combination of virtual network functions (VNFs) and commodity servers. A certain network service can be composed of a sequence of VNFs, i.e., service (function) chain. The service chaining (SC) problem aims to establish an appropriate service path from the origin node to the destination node, which holds both the resource constraints and service chain requirements of executing the required VNFs in the designated order. SC belongs to the complexity class **NP**-hard. In the previous work, inspired by the similarity between the SC problem and the shortest path tour problem (SPTP), we showed the capacitated SPTP (CSPTP) based ILP for the SC problem, where CSPTP is a generalized version of the SPTP with both the node and link capacity constraints. In this paper, we propose Lagrangian heuristics to solve the CSPTP-based ILP for the SC in a speedy and efficient manner. We further present that the proposed heuristics can also solve both the service chaining and function placement by slightly extending the network model called an

[‡]This article is an extended version of the paper to be presented at the 2022 IEEE/IFIP Network Operations and Management Symposium (IEEE NOMS 2022) [1].

augmented network. Through numerical results, we show that the proposed heuristics for the SC is competitive with the optimal resource allocation while executing much faster than the combination of the CSPTP-based ILP and the existing solver, i.e., CPLEX. Furthermore, we also show that the proposed heuristics for both the service chaining and function placement can still balance the solution optimality and computational complexity, thanks to the parallel computation architectures.

Keywords: Network functions virtualization (NFV), service chaining, function placement, integer linear programming (ILP), capacitated shortest path tour problem (CSPTP), Lagrangian relaxation, linear relaxation, subgradient algorithm, totally unimodular

1 Introduction

Network functions virtualization (NFV) can realizes more flexible and diverse network services by liberating network functions from the conventional network equipment (e.g., firewall, network address translation (NAT), and deep packet inspection) and executing them as virtual network functions (VNFs) on commodity servers (e.g., high volume servers) [2–5]. A certain network service can be composed of a sequence of VNFs, called a *service (function) chain* [6].

A *service chaining* (SC) problem, which is one of the resource allocation problems in NFV networks, tries to establish a service chain in response to a *service chain request* (SCR) with its requirements (i.e., origin and destination nodes, a sequence of functions, and demand for bandwidth and processing) [4, 5]. It belongs to the complexity class **NP**-hard [7]. More specifically, the SC problem is finding an appropriate *service path* from the origin node to the destination node while running the VNFs at the intermediate nodes in the required order under the resource constraints. Furthermore, locations of functions in NFV networks affect the performance of service path, which causes another important resource allocation problem called *function placement*.

Recently, several studies [8–10] have pointed out that the SC problem is similar to *shortest path tour problem* (SPTP) [11, 12]. The SPTP is an extended version of the shortest path problem (SPP) and aims to find the shortest path from an origin node to a destination node while traversing at least one intermediate node from given disjoint node subsets $\mathcal{T}_1, \ldots, \mathcal{T}_K$ in this order [13]. Festa proved that the SPTP belongs to the complexity class **P** [13]. Bhat and Rouskas first showed the similarity between SC and SPTP and developed an algorithm for calculating the shortest path tour, called depth first tour search (DFTS) [8].

Different from the SPTP, the SC problem has the node and link capacity constraints. Focusing only on the constraints on link capacities, several studies extended the SPTP to constrained SPTP where the path tour does not traverse an identical link more than once [14–16]. Ferone et al. proved that the constrained SPTP belongs to **NP**-complete [14]. In [15, 16], the constrained SPTP is formulated as an integer linear program (ILP) and solved using Lagrangian heuristics.

In our previous work [10], we revealed that the SC problem can exactly be modeled as a *capacitated* SPTP (CSPTP), which is a general case of the constrained SPTP, where it supports more general capacity constraints on both nodes and links with real values. Inspired by the ILP formulation for the constrained SPTP [15] and a special network model called an *augmented network*, we exactly formulated the SC problem as a CSPTP-based ILP. (We will explain the augmented network and CSPTP-based ILP formulation in Section 3.2 and Section 3.4, respectively.) Furthermore, we extended this ILP for both the service chaining and function placement (SCFP), which can determine the appropriate service path as well as the appropriate number and locations of VNFs in NFV networks. (The details about the CSPTP-based ILP for both the service chaining and function placement will be explained in Section 3.5.)

The high difficulties of CSPTP-based service chaining (and function placement) comes from the following characteristics: (1) ensuring the use of identical links as many times as required, (2) holding sequential execution of VNFs in required order, and (3) meeting the node and link capacity constraints.

From the viewpoint of practicality, many studies proposed heuristic algorithms for solving SC and/or SCFP to overcome the computational complexity, in addition to the formulation of the target problems as optimization problems [9, 10, 17–21]. To deal with the tradeoff between solution optimality and computational complexity, we proposed the DFTS-based Lagrangian heuristics only for SC [1]. In this paper, we generalize the Lagrangian heuristics for both SC and SCFP by integrating existing techniques, i.e., CSPTP, Lagrangian relaxation, linear relaxation, DFTS, and subgradient algorithm. To be more precise, we propose two types of Lagrangian heuristics, i.e., linear program (LP)-based one and DFTS-based one, depending on the way to solve the Lagrangian problem. (The details of the proposed heuristics will be given in Section 4.) Through simulation results, we evaluate the fundamental characteristics of the proposed heuristics and reveal the relationship between optimality of resource allocation and computational complexity.

The main contributions of the manuscript are as follows:

1. To overcome the high computational complexity of the CSPTP while keeping the optimality of SC, we propose the two types of Lagrangian heuristics, i.e., LP-based one and DFTS-based one, for the CSPTP-based SC in a speedy and efficient manner. Although the proposed heuristics is composed of existing techniques, each of which was separately considered in existing studies, its novelty is the integration way of them by taking account of the advantage of each technique. In addition, we reveal that the proposed heuristics can also solve SCFP by slightly extending the augmented network.
2. Inspired by [16], we prove that the Lagrangian problem of the CSPTP-based ILP has a totally unimodular constraint matrix, which guarantees that the

linear relaxation of the Lagrangian problem maintains the integrality of the decision variables (i.e., optimal solution(s)) while reducing the practical computation time.

3. Through numerical results of online SC, we demonstrate that the proposed heuristics can calculate almost optimal service paths even under situations where the original CSPTP-based ILP cannot be solved by the existing solver, CPLEX [22], in a real-time manner. We also show that the proposed heuristics can speedily and efficiently solve SCFP in a batch manner, thanks to the parallel computation architectures.

The rest of the manuscript is organized as follows. Section 2 gives the related work. In Section 3, we give the system model and CSPTP-based ILP for SC and SCFP. To address the computational complexity, we propose the Lagrangian-based heuristics for solving the CSPTP-based ILP for SC and SCFP in Section 4. Sections 5 and 6 demonstrate the fundamental characteristics of the proposed heuristics for SC and SCFP, respectively. Finally, Section 7 gives the conclusions.

2 Related Work

2.1 Service Chaining Problem

SC is one of the challenging resource allocation problems in NFV networks [3–5]. It aims to establish an appropriate service path while holding both the resource constraints and service chain requirements, which belongs to the complexity class **NP**-hard [7]. To address this problem, there have been many studies on efficiently solving the SC problem [10, 17–21]. Sun et al. proposed an ILP formulation and a heuristic algorithm for the power-efficient and traffic-aware SC in multi-domain networks [19]. Huin et al. applied the layered graph model to the ILP formulation for the SC problem to minimize the total link utilization and developed the heuristic algorithm [20]. Nguyen et al. adopted the expanded network model to the ILP formulation for the SC problem to minimize the total network utilization and developed the Lagrangian heuristic algorithms [21]. Sallam et al. applied the graph transformation to the SC problem and formulated it as a multi-commodity maximum flow problem [17]. Recent surveys on the SC problem are found in [3–5].

2.2 Similarity between Service Chaining Problem and SPTP

Several studies pointed out that the SC problem is similar to SPTP [8–10]. Bhat and Rouskas developed the efficient algorithm for calculating the SPTP, called DFTS [8]. The DFTS algorithm, however, does not consider the load and capacity of each physical node and link. Focusing on the similarity between SC and SPTP, Gao et al. implemented the congestion-aware online routing algorithm for SC to minimize the maximum network utilization by using a potential function [9]. Similarly, in [10], we formulated the CSPTP-based ILP for SC using an augmented network and developed a greedy-based heuristic algorithm,which sequentially conducts the shortest path selection. To improve the balance between computational complexity and optimality, in this paper, we propose the DFTS and LP-based Lagrangian heuristics for the online SC to solve the CSPTP with low computational complexity. The difference between the DFTS-based one and LP-based one is the way to solve the Lagrangian problem. The first one employs the existing shortest path tour algorithm [8] while the second one focuses on the fact that the Lagrangian problem of the original CSPTP-based ILP can be linearly relaxed without losing its optimality. We further present the applicability of the proposed heuristics to SCFP.

As mentioned in Section 1, the CSPTP is the generalized version of the constrained SPTP [14–16]. Saraiva and Andrade formulated the constrained SPTP as an ILP and implemented the Lagrangian-based heuristic algorithm [15, 16]. In [16], they also showed that the constraint matrix of the Lagrangian problem of the constrained SPTP-based ILP is totally unimodular [23], which guarantees that feasible solution(s) are located at extreme point(s) of an integral polyhedron. In the conference version of this paper [1], we proposed the DFTS-based Lagrangian heuristics for SC. In this paper, we further propose the LP-based Lagrangian heuristics for SC, which guarantees the linear relaxation of the Lagrangian heuristics maintains the integrality of the decision variable while reducing the practical computation time, thanks to the totally unimodular constraint matrix. Inspired by [16], we will show that the Lagrangian problem of our CSPTP-based ILP for SC also has the totally unimodular constraint matrix.

2.3 Service Chaining and Function Placement Problem

Service paths require to visit some physical nodes capable of their demanding functions (VNFs). In other words, the function placement problem also arises to determine the optimal number and locations of functions, which can yield optimal service paths while suppressing the deployment costs. Bhamare et al. formulated the VNF placement problem as an ILP to minimize the response time and inter-cloud traffic and proposed a greedy based heuristic approach [24]. In [25], the authors modeled the VNF placement as a facility location problem to minimize resource consumption and proposed a heuristic algorithm to tackle the computational complexity. Hyodo et al. estalibshed the function placement problem for SC as an ILP thanks to a layered graph and developed a heuristic algorithm by relaxing both visiting order and non-loop constraints [18].

Solving both the service chaining and function placement at the same time will open the possibility of better service paths at the cost of further computational complexity [10, 26–28]. Dieye et al. formulated SCFP in content delivery networks as an ILP to minimize the operational cost while meeting a certain level of quality of service (QoS), i.e., the predefined delay threshold, and proposed a heuristic algorithm based on the Page Rank algorithm [26]. In [27], the authors proposed an ILP for both the online and batch SCFP to minimize the resource consumption considering the sharing of VNFs across tenants. Alleg et al. formulated a mixed integer quadratically constrained program for SCFP to minimize the resource consumption while satisfying service level agreements, i.e., the predefined end-to-end delay threshold [28]. Kiji et al. formulated a VNF placement and routing model for multicast services as an ILP to minimize the placement cost and link utilization [29, 30].

In [10], we also dealt with the CSPTP-based ILP for SCFP and proposed a simple greedy-based heuristic algorithm to tackle the scalability and computational complexity. In the conference version of this paper [1], we tackled the computational complexity of the SC problem with the help of the DFTS-based Lagrangian heuristics only for SC. In addition to the SC problem, in this paper, we further propose the DFTS/LP-based Lagrangian heuristics for SCFP by extending the augmented network model proposed in the previous work [10], which can determine both the optimal number and locations of functions by finding service paths while exploring all possibilities of function placement.

2.4 Network Models for Service Chaining

The above mentioned special network models, i.e., layered graph [18, 20], expanded network [21], and augmented network [10], contribute to achieving effective SC. Both the layered graph and expanded network generate a hierarchical network with multiple copies (layers) of the original physical network in response to the service chain requirement [18, 20, 21]. This indicates that they may be required to reconstruct the hierarchical network per SCR because each SCR may require the different number of VNFs and/or different VNF sequence.

Different from them, the augmented network can be constructed by extending the original physical network with imaginary nodes, each of which is responsible for the corresponding VNF and connected to physical node(s) possessing it through a virtual link. Note that the details of the augmented network will be shown in Section 3.2. The augmented network can support arbitrary SCRs by preparing the imaginary nodes for all possible functions. This leads to alleviate the overhead of the network reconstruction and will be more attractive to the online SC. Therefore, in this paper, we adopt the augmented entwork to solve the online SC.

The augmented network approach [10] is similar to the virtual network embedding (VNE) problem [31] in terms of the network construction way. More precisely, the augmented network approach aims at mapping a function f to a physical node v by connecting the physical node v to the imaginary node $\hat{v}_f$ responsible for the function f through the virtual link $(\hat{v}_f, v)$. On the other hand, the VNE problem aims at mapping a virtual node (resp. a virtual link) to a physical node (resp. physical link(s)). We should note that the term "virtual link" has a different meaning between the two approaches. In other words, the augmented network approach considers only function mapping while the VNE problem considers both node and link mapping.

Table 1 Notations.

Symbol	Description
G	Physical network $G = (\mathcal{V}, \mathcal{E})$
$\mathcal{V}$	Set of physical nodes, $V = \vert\mathcal{V}\vert$
$\mathcal{V}_{\text{VNF}}$	Set of VNF-enabled physical nodes, $\mathcal{V}_{\text{VNF}} \subseteq \mathcal{V}$, $V_{\text{VNF}} = \vert\mathcal{V}_{\text{VNF}}\vert$
$\mathcal{E}$	Set of physical links, $E = \vert\mathcal{E}\vert$
$\mathcal{F}$	Set of functions, $\mathcal{F} = \{f_1, \ldots, f_F\}$, $F = \vert\mathcal{F}\vert$
$\mathcal{F}_i$	Set of functions contained in physical node $i \in \mathcal{V}$
$\mathcal{N}_f$	Set of physical nodes possessing function $f \in \mathcal{F}$ in SC ($N_f = \vert\mathcal{N}_f\vert$)
$\mathcal{N}_f^*$	Set of physical nodes possessing function $f \in \mathcal{F}$ in SCFP ($N_f^* = \vert\mathcal{N}_f^*\vert$)
$\mathcal{C}$	Set of SCRs, $C = \vert\mathcal{C}\vert$
c	Service chain request with origin o_c and destination d_c
$\mathcal{R}_c$	Sequence of functions $(f_{c,1}, \ldots, f_{c,K_c})$ required by c
$\mathcal{K}_c$	Set of function indices required by c, $\mathcal{K}_c = \{1, \ldots, K_c\}$
$\mathcal{K}_c^+$	Set of subpath indices of $\mathcal{S}_c$, $\mathcal{K}_c = \{1, \ldots, K_c + 1\}$
G^+	Augmented network $G = (\mathcal{V}^+, \mathcal{E}^+)$,
$\mathcal{V}^+$	Set of nodes on augmented network, $\mathcal{V}^+ = \mathcal{V} \cup \hat{\mathcal{V}}$
$\mathcal{E}^+$	Set of links on augmented network, $\mathcal{E}^+ = \mathcal{E} \cup \hat{\mathcal{E}}^{\text{in}} \cup \hat{\mathcal{E}}^{\text{out}}$
$\hat{\mathcal{V}}$	Set of imaginary nodes, $\hat{\mathcal{V}} = \{\hat{v}_f\}_{f \in \mathcal{F}}$, where imaginary node $\hat{v}_f$ is responsible for function f
$\mathcal{V}_i^+$	Set of neighbors of node i, $\mathcal{V}_i^+ \subseteq \mathcal{V}^+$
$\hat{\mathcal{E}}^{\text{in}}$	Set of incoming virtual links, $\hat{\mathcal{E}}^{\text{in}} = \{(v, \hat{v}) \mid v \in \mathcal{V}, \hat{v}_f \in \hat{\mathcal{V}}, f \in \mathcal{F}_v\}$
$\hat{\mathcal{E}}^{\text{out}}$	Set of outgoing virtual links, $\hat{\mathcal{E}}^{\text{out}} = \{(\hat{v}, v) \mid \hat{v}_f \in \hat{\mathcal{V}}, v \in \mathcal{V}, f \in \mathcal{F}_v\}$
$\mathcal{S}_c$	Service path for c, $(\mathcal{S}_{c,1}, \ldots, \mathcal{S}_{c,K_c+1})$
$\mathcal{S}_{c,k}$	kth sub service path with origin $\alpha_{c,k}$ and destination $\beta_{c,k}$
b_c	Bandwidth requirement of c
p_c^{node}	Processing requirement of c for traversing a node
$p_{c,f_{c,k}}^{\text{func}}$	Processing requirement of $f_{c,k} \in \mathcal{R}_c$ at a node
$B_{i,j}$	Residual bandwidth of link (i, j) at arrival of c
P_i	Residual processing capacity of node i at arrival of c
$d_{i,j}^{\text{link}}$	Propagation delay of link $(i, j) \in \mathcal{E}$
d_i^{node}	Traversal delay of node $i \in \mathcal{V}$
$d_{\hat{v}_f,v}^{\text{func}}$	Processing delay of function f of node $v \in \mathcal{V}$
$x_{i,j}^{c,k}$	Binary decision variables: 1: if link (i, j) is included in $\mathcal{S}_{c,k}$, 0: otherwise
$y_i^{c,k}$	Binary decision variables: 1: if node i is used in $\mathcal{S}_{c,k}$, 0: otherwise
$\mathcal{T}^{c,k}$	Set of imaginary nodes responsible for function $f_{c,k} \in \mathcal{R}_c$
$\boldsymbol{\mu}$, $\boldsymbol{\gamma}$	Lagrangian multipliers
$\boldsymbol{h}$, $\boldsymbol{g}$	Subgradient
ω, θ, ε	Weighting parameter, step size, and optimality tolerance
τ, $T_{\max}$	Iteration ID and maximum number of iterations in Lagrangian heuristics
$Z_{\text{SC}}(\boldsymbol{x})$	CSPTP-based ILP for SC
$Z_{\text{SCFP}}(\boldsymbol{x})$	CSPTP-based ILP for SCFP
$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\gamma})$	Lagrangian function
$\Phi_{\text{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma})$	Lagrangian problem for SCFP
$\Phi_{\text{SCFP}}(\boldsymbol{\mu}, \boldsymbol{\gamma})$	Lagrangian problem for SCFP

3 System Model

We consider the system model used in [10]. In this section, we briefly explain it from the viewpoint of the SCR, the augmented network, and the CSPTP-based ILP for the service chaining (and function placement), respectively. Table 1 summarizes the notations used in this paper.

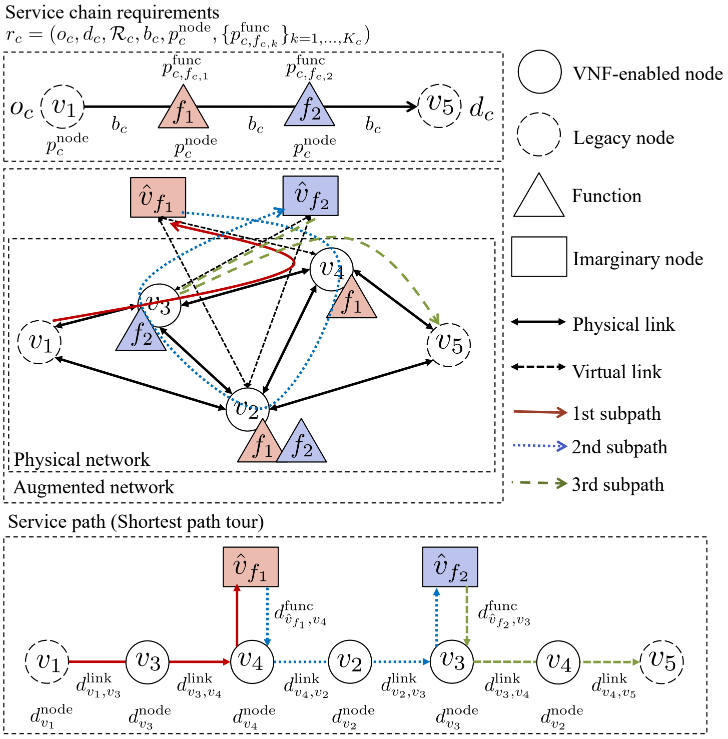

Fig. 1 Overview of SC.

3.1 Service Chain Request

We assume that SCRs arrive at the NFV network, e.g., a carrier network, following a certain stochastic process, e.g., Poisson process. An NFV orchestrator waits for C ($C \geq 1$) SCRs and solves the service chaining (and function placement) problem for the pooled requests $\mathcal{C}$. The processing way of the SCRs changes according to the service type and can be categorized into two types, i.e., online or batch processing. In the online processing ($C = 1$), a new arriving SCR c will be immediately served by the orchestrator to realize the network service with high real-time property. On the contrary, in the batch processing ($C \geq 2$), the NFV orchestrator collectively solves the service chaining (and function placement) problem every C requests.

We assume an SCR c with service chain requirements $r_c = (o_c, d_c, \mathcal{R}_c, b_c, p_c^{\text{node}}, \{p_{c,f_{c,k}}^{\text{func}}\}_{k=1,\dots,K_c})$. o_c and d_c denote an origin node and a destination node, respectively. $\mathcal{R}_c$ is a sequence of K_c functions in the required order, i.e., $(f_{c,1}, \dots, f_{c,K_c})$. b_c denotes the required constant bit rate. p_c^{node} (resp. $p_{c,f_{c,k}}^{\text{func}}$) represents the processing capacity required for forwarding packets (resp. executing the kth function $f_{c,k}$) at a physical node. We illustrate an example of the service chain requirements for the SCR c with two functions, f_1 and f_2, at the top layer of Fig. 1. Note that $f_{c,1} = f_1$ and $f_{c,2} = f_2$.

3.2 Network

We consider a directed graph $G = (\mathcal{V}, \mathcal{E})$ describing a physical network, where $\mathcal{V}$ (resp. $\mathcal{E}$) denotes a set of physical nodes (resp. links). Each physical link $(i, j) \in \mathcal{E}$ (resp. each physical node $i \in \mathcal{V}$) has residual bandwidth $B_{i,j}$ (resp. residual processing capacity P_i) at the beginning of serving the SCRs $\mathcal{C}$. The NFV network supports a set of F distinct network functions, $\mathcal{F} = \{f_1, \ldots, f_F\}$, and consists of two types of physical nodes: VNF-enabled physical nodes $\mathcal{V}_{\text{VNF}}$ ($V_{\text{VNF}} = |\mathcal{V}_{\text{VNF}}|$) and legacy ones (i.e., routers and switches). Each VNF-enabled node $i \in \mathcal{V}_{\text{VNF}}$ can run one or more functions $\mathcal{F}_i \subseteq \mathcal{F}$ while the legacy one only supports data forwarding. Since the deployment of function(s) to physical nodes will consume network bandwidth and storage capacity, we assume that the network administrator wants to deploy each function $f \in \mathcal{F}$ to a minimum number of VNF-enabled physical nodes, i.e., $\mathcal{N}_f \subseteq \mathcal{V}_{\text{VNF}}$ ($N_f = |\mathcal{N}_f|$). Note that N_f VNF-enabled physical nodes are predetermined in case of SC while the number N_f^* and set (locations) $\mathcal{N}_f^*$ of VNF-enabled physical nodes possessing function $f \in \mathcal{F}$ will be dynamically adjusted in case of SCFP.

In Fig. 1, the physical network is given at the middle layer and supports two functions ($\mathcal{F} = \{f_1, f_2\}$). The physical nodes v_2, v_3, and v_4 are VNF-enabled physical nodes, each of which possesses one or two functions, i.e., $\mathcal{F}_{v_2} = \{f_1, f_2\}$, $\mathcal{F}_{v_3} = \{f_2\}$, and $\mathcal{F}_{v_4} = \{f_1\}$. From the viewpoint of each function, $\mathcal{N}_{f_1} = \{v_2, v_4\}$ and $\mathcal{N}_{f_2} = \{v_2, v_3\}$. Please note that f_1, which is required as $f_{c,1}$, is possessed by two physical nodes v_2 and v_4 but it will be executed only at v_4 in this case. (The details will be explained later.)

To deal with the CSPTP, we extend the physical network G to an augmented network $G^+ = (\mathcal{V}^+, \mathcal{E}^+)$ with imaginary nodes $\hat{\mathcal{V}}$ and virtual links $\hat{\mathcal{E}}^{\text{in}} \cup \hat{\mathcal{E}}^{\text{out}}$. Note that $\mathcal{V}^+ = \mathcal{V} \cup \hat{\mathcal{V}}$ and $\mathcal{E}^+ = \mathcal{E} \cup \hat{\mathcal{E}}^{\text{in}} \cup \hat{\mathcal{E}}^{\text{out}}$. An imaginary node $\hat{v}_{f_{c,k}} \in \hat{\mathcal{V}}$ is in charge of the kth function $f_{c,k}$, which is connected to each physical node possessing $f_{c,k}$ through a virtual link. $\hat{\mathcal{E}}^{\text{in}}$ (resp. $\hat{\mathcal{E}}^{\text{out}}$) denotes a set of virtual links incoming to (resp. outgoing from) an imaginary node $\hat{v}_f$. Note that $\hat{\mathcal{E}}^{\text{in}} = \{(v, \hat{v}_f) \mid v \in \mathcal{V}_{\text{VNF}}, \hat{v}_f \in \hat{\mathcal{V}}, f \in \mathcal{F}_v\}$ (resp. $\hat{\mathcal{E}}^{\text{out}} = \{(\hat{v}_f, v) \mid \hat{v}_f \in \hat{\mathcal{V}}, v \in \mathcal{V}_{\text{VNF}}, f \in \mathcal{F}_v\}$). The virtual link $(\hat{v}_{f_{c,k}}, v) \in \hat{\mathcal{E}}^{\text{out}}$ means that the physical node $v \in \mathcal{N}_{f_{c,k}}$ possesses the function $f_{c,k}$. Furthermore, selecting the virtual link $(\hat{v}_{f_{c,k}}, v)$ as a part of the service path indicates the execution of the function $f_{c,k}$ at the physical node v. Let $\mathcal{V}_i$ denote a set of neighbors of node i. In Fig. 1, we show the augmented network at the middle layer. In this example, the outgoing virtual link $(\hat{v}_{f_1}, v_4)$ (resp. $(\hat{v}_{f_2}, v_3)$) is selected as a part of the service path, which means that f_1 (resp. f_2) will be executed at v_4 (resp. v_3).

In contrast to the existing network models [18, 20, 21], i.e., the layered graph and expanded network, the augmented network is constructed only at once, which can support arbitrary SCRs by preparing the imaginary nodes for all the functions $\mathcal{F}$. As a result, the augmented network can alleviate the overhead of network reconstruction.

3.3 Service Path

A service path $\mathcal{S}_c$ with the origin o_c, destination d_c, and $R_c = (f_{c,1}, \ldots, f_{c,K_c})$ is composed of a sequence of $K_c + 1$ subpaths, i.e., $(\mathcal{S}_{c,1}, \ldots, \mathcal{S}_{c,K_c+1})$. The pair $(\alpha_{c,k},\ \beta_{c,k})$ of origin and desitnation nodes of the kth subpath is given as follows:

$$(\alpha_{c,k}, \beta_{c,k}) = \begin{cases} (o_c, \hat{v}_{f_{c,1}}), & k = 1, \\ (\hat{v}_{f_{c,k-1}}, \hat{v}_{f_{c,k}}), & k = 2, \ldots, K_c, \\ (\hat{v}_{f_{c,K_c}}, d_c), & k = K_c + 1. \end{cases}$$

For example, in the bottom layer of Fig. 1, the entire service path $\mathcal{S}_c = (v_1, v_3, v_4, \hat{v}_{f_1}, v_4, v_2, v_3, \hat{v}_{f_2}, v_3, v_2, v_5)$ is decomposed into three subpaths, i.e., $\mathcal{S}_{c,1} = (v_1, v_3, v_4, \hat{v}_{f_1})$, $\mathcal{S}_{c,2} = (\hat{v}_{f_1}, v_4, v_2, v_3, \hat{v}_{f_2})$, and $\mathcal{S}_{c,3} = (\hat{v}_{f_2}, v_3, v_2, v_5)$. We can confirm that executing the function f_1 (resp. f_2) at the physical node v_4 (resp. v_3) corresponds to selecting the virtual link $(\hat{v}_{f_1}, v_4)$ (resp. $(\hat{v}_{f_2}, v_3)$) as part of the service path $\mathcal{S}_c$. Each subpath does not include any loop but the entire service path may have loop(s). In this example, we can observe that the physical link (v_3, v_4) is used twice in the service path S_c while it is used in each subpath at most once. We cannot know how many times one link will be included in the service path before calculating the service path itself, which is one of the reasons making the SC problem **NP**-complete. We evaluate the optimality of a service path by total delay, which will be defined in Section 3.4.

3.4 CSPTP-Based ILP for Service Chaining

Thanks to the augmented network, we formulate the SC problem as the following ILP $\mathrm{Z}_{\mathrm{SC}}(\boldsymbol{x})$. Let $\boldsymbol{x} = [x_{i,j}^{c,k}]$ $(c \in \mathcal{C}, k \in \mathcal{K}_c^+, (i,j) \in \mathcal{E}^+)$ denote the binary decision variables [10]:

$$x_{i,j}^{c,k} = \begin{cases} 1, & \text{if a physical/virtual link } (i,j) \text{ is included in} \\ & k\text{th subpath of a service path for SCR } c, \\ 0, & \text{otherwise.} \end{cases}$$

$$\min_{\boldsymbol{x}} \quad \sum_{c \in \mathcal{C}} \sum_{(i,j) \in \mathcal{E}^+} d_{i,j} \sum_{k \in \mathcal{K}_c^+} x_{i,j}^{c,k}, \tag{1}$$

$$\text{s.t.} \quad x_{i,j}^{c,k} = \{0, 1\}, \qquad (i,j) \in \mathcal{E}^+, c \in \mathcal{C}, k \in \mathcal{K}_c^+, \tag{2}$$

$$\sum_{j \in \mathcal{V}_i^+} x_{i,j}^{c,k} - \sum_{j \in \mathcal{V}_i^+} x_{j,i}^{c,k} = \begin{cases} 1 & \text{if } i = \alpha_{c,k}, \\ -1 & \text{if } i = \beta_{c,k}, \\ 0 & \text{otherwise,} \end{cases}$$

$$i \in \mathcal{V}^+, c \in \mathcal{C}, k \in \mathcal{K}_c^+, \tag{3}$$

$$x_{i,\hat{v}_{f_{c,k}}}^{c,k} = x_{\hat{v}_{f_{c,k}},i}^{c,k+1},$$

$$(i, \hat{v}_{f_{c,k}}) \in \hat{\mathcal{E}}^{\text{in}}, (\hat{v}_{f_{c,k}}, i) \in \hat{\mathcal{E}}^{\text{out}}, c \in \mathcal{C}, k \in \mathcal{K}_c, \tag{4}$$

$$x^{c,k}_{i,\hat{v}_{f_{c,m}}} = 0,$$

$$(i, \hat{v}_{f_{c,m}}) \in \hat{\mathcal{E}}^{\text{in}}, c \in \mathcal{C}, k \in \mathcal{K}^+_c, m \neq k, \tag{5}$$

$$\sum_{c \in \mathcal{C}} \big(b_c \sum_{k \in \mathcal{K}^+_c} x^{c,k}_{i,j}\big) \leq B_{i,j}, \qquad (i,j) \in \mathcal{E}, \tag{6}$$

$$\sum_{c \in \mathcal{C}} \big(p^{\text{node}}_c \sum_{(v,j) \in \mathcal{E}} \sum_{k \in \mathcal{K}^+_c} x^{c,k}_{v,j} + \sum_{(\hat{v}_f, v) \in \hat{\mathcal{E}}^{\text{out}}} p^{\text{func}}_{c,f} \sum_{k \in \mathcal{K}^+_c} x^{c,k}_{\hat{v}_f, v}\big) \leq P_v,$$

$$v \in \mathcal{V}. \tag{7}$$

The objective function (1) represents the minimization of the total delay of the service path, where $d_{i,j}$ is defined as follows.

$$d_{i,j} = \begin{cases} d^{\text{node}}_i + d^{\text{link}}_{i,j}, & \text{if } (i,j) \in \mathcal{E}, \\ d^{\text{func}}_{i,j}, & \text{if } (i,j) \in \hat{\mathcal{E}}^{\text{out}}, \\ 0, & \text{otherwise.} \end{cases}$$

The total path delay is the sum of propagation delay and processing delay along the path, where each physical link (i,j) requires the propagation delay $d^{\text{link}}_{i,j}$ and each physical node $v \in \mathcal{V}$ requires the processing delay d^{node}_v for data forwarding. The VNF-enabled physical node $v \in \mathcal{V}_{\text{VNF}}$ possessing function $f \in \mathcal{R}_c$ also has the processing delay $d^{\text{func}}_{\hat{v}_f,v}$ for executing the function f.

Constraint (2) gives the domain of the binary decision variables. The CSPTP-related constraints are given by (3)–(7) where constraints (3)–(5) are related to the SPTP. Constraint (3) guarantees the standard flow conservation rules. Constraint (4) ensures the connectivity between kth and $(k+1)$th subpaths of SCR c. Constraint (5) restricts the unnecessary execution of mth function $f_{c,m}$ in the kth $(m \neq k)$ subpath. Constraints (6) and (7) give the constraints on physical link capacity and processing capacity of the physical node, respectively.

3.5 CSPTP-Based ILP for Service Chaining and Function Placement

The CSPTP-based ILP for SCFP, $\mathrm{Z}_{\text{SCFP}}(\boldsymbol{x})$, can similarly be formulated by extending the augmented network as follows. In the augmented network for SCFP, we connect each imaginary node, i.e., function, to all VNF-enabled physical nodes through virtual links, which considers all possibilities of function placement. More specifically, the set of incoming (resp. outgoing) virtual links is updated as $\hat{\mathcal{E}}^{\text{in}} = \{(v, \hat{v}_f) \mid v \in \mathcal{V}_{\text{VNF}}, \hat{v}_f \in \hat{\mathcal{V}}, f \in \mathcal{F}\}$ (resp. $\hat{\mathcal{E}}^{\text{out}} = \{(\hat{v}_f, v) \mid \hat{v}_f \in \hat{\mathcal{V}}, v \in \mathcal{V}_{\text{VNF}}, f \in \mathcal{F}\}$). Note that the actual number N^*_f and location(s) of each function $f \in \mathcal{F}$ will be determined by solving $\mathrm{Z}_{\text{SCFP}}(\boldsymbol{x})$.

Fig. 2 illustrates the comparison between the augmented network for SC and that for SCFP. The left bottom figure shows the augmented network for

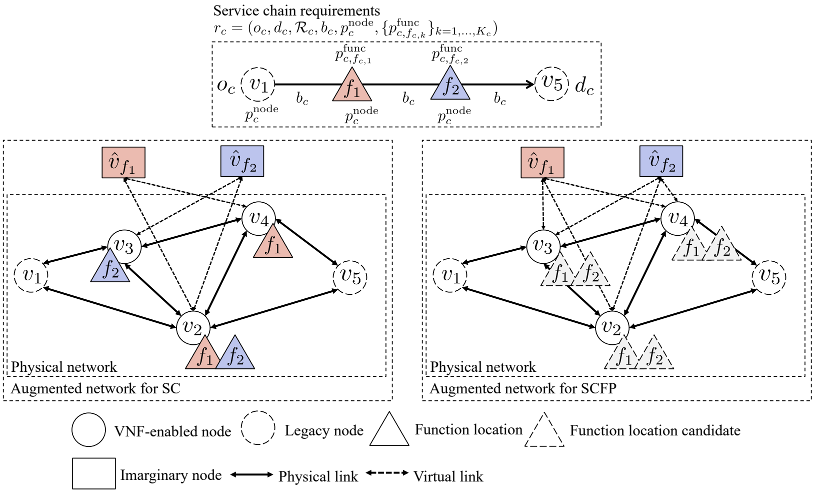

Fig. 2 Comparison between the augmented network for SC and that for SCFP.

SC, which is the same as that in Fig. 1. In this example, the locations of f_1 (resp. f_2) are predefined to be v_2 and v_4 (resp. v_2 and v_3). On the other hand, the right bottom figure is the augmented network for SCFP. We confirm that all the VNF-enabled physical nodes (v_2, v_3, and v_4) become the location candidates of two functions f_1 and f_2.

3.6 Service Chaining vs Service Chaining with Function Placement

Finally, we discuss the relationship between SC and SCFP from the viewpoint of the tradeoff between the speediness and adaptability of service provisioning. SC tries to find an optimal service path under the predefined function locations, and thus it can speedily construct the service path and suitable for network services with high real-time property, i.e., online processing. However, if the network load increases, it may fail in accepting future SCRs, due to lack of resources.

On the contrary, SCFP can flexibly adapt to demand change at the increasing cost of computation and deployment. Therefore, it is suitable for batch processing, which can utilize the network resources more effectively and accept more SCRs under a certain predefined processing time limit. More specifically, the batch processing simultaneously serves multiple SCRs with a certain batch size, which can alleviate the myopic resource allocation.

4 Lagrangian Heuristics for CSPTP-Based ILP for Service Chaining and Function Placement

To overcome the computational complexity as well as maintaining the solution optimality as much as possible, we propose Lagrangian heuristics to solve

Algorithm 1 Lagrangian heuristics to solve CSPTP-based ILP for service chaining (and function placement).

```
Require: Optimality tolerance $\varepsilon$, maximum number $T_{\max}$ of iterations, and weighting parameter $\omega$.
Ensure: Solution $\boldsymbol{x}^*$.
$\tau \leftarrow 0,\ \boldsymbol{\mu}^\tau \leftarrow \mathbf{0},\ \boldsymbol{\gamma}^\tau \leftarrow \mathbf{0}$
do
    $(\boldsymbol{x}^*, \mathcal{L}^\tau) \leftarrow \textsc{solve}(\Phi_{\mathrm{SC}}(\boldsymbol{\mu}^\tau, \boldsymbol{\gamma}^\tau))$
    if $\boldsymbol{x}^*$ is infeasible then
        return None
    end if
    $(\boldsymbol{h}^\tau, \boldsymbol{g}^\tau) \leftarrow \textsc{subgradient}(\Phi_{\mathrm{SC}}(\boldsymbol{\mu}^\tau, \boldsymbol{\gamma}^\tau))$
    $\theta^\tau_{\boldsymbol{\mu}} \leftarrow \omega/(\sqrt{\tau}||\boldsymbol{h}^\tau||),\ \theta^\tau_{\boldsymbol{\gamma}} \leftarrow \omega/(\sqrt{\tau}||\boldsymbol{g}^\tau||)$
    $\boldsymbol{\mu}^{\tau+1} \leftarrow \max\{0, \boldsymbol{\mu}^\tau + \theta^\tau_{\boldsymbol{\mu}} \boldsymbol{h}^\tau\}$
    $\boldsymbol{\gamma}^{\tau+1} \leftarrow \max\{0, \boldsymbol{\gamma}^\tau + \theta^\tau_{\boldsymbol{\gamma}} \boldsymbol{g}^\tau\}$
    $\tau \leftarrow \tau + 1$
while $\textsc{stop\_condition}()$
return $\boldsymbol{x}^*$
```

the CSPTP-based ILP for SC and SCFP. In what follows, for simplicity of explanation, we mainly focus on SC in Sections 4.1 through 4.5 and explain the extension of the proposed heuristics to SCFP in Section 4.6. Finally, we give the discussion about computational complexity in Section 4.7.

4.1 Overview

Algorithm 1 shows the proposed Lagrangian heuristics, which is a combination of several existing techniques, i.e., Lagrangian relaxation, linear relaxation, the DFTS algorithm for finding the shortest path tour [8], and the subgradient algorithm. We propose two types of the Lagrangian heuristics: *LP-based heuristics* and *DFTS-based heuristics*, according to the way to solve the Lagrangian problem.

We first formulate the Lagrangian dual problem for the CSPTP-based ILP $\mathrm{Z}_{\mathrm{SC}}(\boldsymbol{x})$ applying the Lagrangian relaxation, which transforms CSPTP into SPTP. (The details will be given in Section 4.2.) Next, we develop the two approaches to efficiently solve the Lagrangian problem. In the first approach, we further reformulate the Lagrangian problem and apply the linear relaxation to it, which will be given in Section 4.3. In the second approach, thanks to the Lagrangian relaxation, we adopt the DFTS to realize the efficient computational complexity. (The details will be shown in Section 4.4.) Since both approaches require to cope with the original capacity constraints, we also adopt the subgradient algorithm, which will be explained in Section 4.5.

4.2 Lagrangian Relaxation

As mentioned in Section 1, the constrained SPTP only takes into account the link capacity constraints and belongs to **NP**-complete [14]. Since the CSPTP takes account of both the node and link capacity constraints, i.e., (6) and (7), it would be also expected to belong to **NP**-complete. Because the SPTP belongs to the lower complexity class **P** [13], we transform the CSPTP-based ILP $\mathrm{Z}_{\mathrm{SC}}(\boldsymbol{x})$ into the SPTP-based ILP, i.e., Lagrangian problem, with the help of Lagrangian relaxation. We formulate the Lagrangian function $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\gamma})$ by relaxing constraints (6) and (7) with multipliers $\boldsymbol{\gamma} = (\gamma_{i,j})_{(i,j)\in\mathcal{E}}$ and $\boldsymbol{\mu} = (\mu_v)_{v\in\mathcal{V}}$ $(\gamma_{i,j} \geq 0, \mu_v \geq 0)$, respectively:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\gamma}) = & \sum_{c\in\mathcal{C}} \sum_{(i,j)\in\mathcal{E}} \sum_{k\in\mathcal{K}_c^+} \left(d_{i,j} + \mu_i p_c^{\mathrm{node}} + \gamma_{i,j} b_c\right) x_{i,j}^{c,k} \\
& + \sum_{c\in\mathcal{C}} \sum_{(\hat{v}_f,v)\in\hat{\mathcal{E}}^{\mathrm{out}}} \sum_{k\in\mathcal{K}_c^+} \left(d_{\hat{v}_f,v}^{\mathrm{func}} + \mu_v p_{c,f}^{\mathrm{func}}\right) x_{\hat{v}_f,v}^{c,k} \\
& - \Bigg(\sum_{(i,j)\in\mathcal{E}} \gamma_{i,j} B_{i,j} + \sum_{i\in\mathcal{V}} \mu_i P_i\Bigg).
\end{aligned}$$

We also formulate the Lagrangian problem $\Phi_{\mathrm{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma})$ as follows:

$$\begin{aligned}
\Phi_{\mathrm{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma}) = \min_{\boldsymbol{x}} \quad & \mathcal{L}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\gamma}) \\
\text{s.t.} \quad & (2)\text{–}(5).
\end{aligned}$$

Since all the constraints (2)–(5) give the SPTP-related ones, $\Phi_{\mathrm{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma})$ belongs to the class **P**. For any $\boldsymbol{\mu} \geq \mathbf{0}$ and $\boldsymbol{\gamma} \geq \mathbf{0}$, $\Phi_{\mathrm{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma})$ gives the lower bound of the optimal objective value of the original CSPTP-based ILP, i.e., $\Phi_{\mathrm{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma}) \leq \mathrm{Z}_{\mathrm{SC}}(\boldsymbol{x})$. Accordingly, finding $\boldsymbol{\mu}$ and $\boldsymbol{\gamma}$ to maximize $\Phi_{\mathrm{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma})$ leads to the minimization of $\mathrm{Z}_{\mathrm{SC}}(\boldsymbol{x})$, which can be derived by solving the following Lagrangian dual problem:

$$\max_{\boldsymbol{\gamma}\geq\mathbf{0}, \boldsymbol{\mu}\geq\mathbf{0}} \quad \Phi_{\mathrm{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma}). \tag{8}$$

4.3 Linear Relaxation

We first focus on the possibility of the linear relaxation of the ILP-based Lagrangian problem $\Phi_{\mathrm{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma})$ while maintaining the integrality of the decision variables (i.e., optimal solution(s)). Although the relaxed LP is still **NP**-complete, in practical, we can much easily solve it compared with the original ILP. To this end, we slightly modify the augmented network to ensure the connectivity between kth and $(k+1)$th subpaths $(k = 1, \ldots, K_c)$. More specifically, we update the set of imaginary nodes as $\hat{\mathcal{V}} = \{\hat{v}_{f,i}\}_{f\in\mathcal{F}, i\in\mathcal{N}_f}$ where each imaginary node represents the pair of function and physical node possessing the function. The corresponding set of incoming virtual links, $\hat{\mathcal{E}}^{\mathrm{in}}$, and

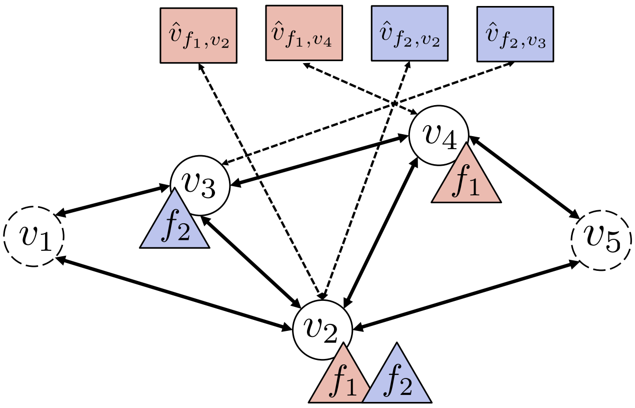

Fig. 3 An example of transformation of augmented network in Fig. 1.

that of outgoing virtual links, $\hat{\mathcal{E}}^{\text{out}}$, are also updated as $\{(i, \hat{v}_{f,i})\}_{f\in\mathcal{F}, i\in\mathcal{N}_f}$ and $\{(\hat{v}_{f,i}, i)\}_{f\in\mathcal{F}, i\in\mathcal{N}_f}$, respectively. Fig. 3 illustrates an example of transformation of the augmented network in Fig.1. Here, we also define a set of imaginary nodes responsible for function $f_{c,k} \in \mathcal{R}_c$ as $\mathcal{T}^{c,k}$ $(c \in \mathcal{C}, k \in \mathcal{K}_c)$. For simplicity of notation, we also define $\mathcal{T}^{c,0} = \{o_c\}$ and $\mathcal{T}^{c,K_c+1} = \{d_c\}$.

Inspired by [16], we reformulate the Lagrangian problem $\Phi_{\text{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma})$ as follows:

$$\min_{\boldsymbol{x}} \quad \mathcal{L}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\gamma})$$

$$\text{s.t.} \quad x_{i,j}^{c,k} = \{0,1\}, \qquad (i,j) \in \mathcal{E}^+, c \in \mathcal{C}, k \in \mathcal{K}_c^+, \tag{9}$$

$$y_i^{c,k} = \{0,1\}, \quad i \in \mathcal{T}^{c,k}, c \in \mathcal{C}, k \in \mathcal{K}_c^+ \cup \{0\}, \tag{10}$$

$$\sum_{j\in\mathcal{V}_i^+} x_{i,j}^{c,k} - \sum_{j\in\mathcal{V}_i^+} x_{j,i}^{c,k} = \begin{cases} y_i^{c,k-1} & \text{if } i \in \mathcal{T}^{c,k-1}, \\ -y_i^{c,k} & \text{if } i \in \mathcal{T}^{c,k}, \\ 0 & \text{otherwise}, \end{cases}$$
$$i \in \mathcal{V}^+, c \in \mathcal{C}, k \in \mathcal{K}_c^+, \tag{11}$$

$$\sum_{i\in\mathcal{T}^{c,k}} y_i^{c,k} = 1, \qquad c \in \mathcal{C}, k \in \mathcal{K}_c, \tag{12}$$

$$y_{o_c}^{c,0} = 1, \qquad c \in \mathcal{C}, \tag{13}$$

$$y_{d_c}^{c,K_c+1} = 1, \qquad c \in \mathcal{C}. \tag{14}$$

Constraints (9) and (10) give the domain of the binary decision variables. The variables $\boldsymbol{y} = [y_i^{c,k}]$ $(c \in \mathcal{C}, k \in \mathcal{K}_c^+ \cup \{0\}, i \in \mathcal{T}^{c,k})$ are defined as

$$y_i^{c,k} = \begin{cases} 1, & \text{if an imaginary node } i \text{ is included in } k\text{th subpath} \\ & \text{of a service path for SCR } c, \\ 0, & \text{otherwise,} \end{cases}$$

and prohibit the service path from visiting the imaginary node $\hat{v}_{i,f_{c,m}}$ in kth subpath $(m \neq k)$. All the constraints (11)–(14) give the SPTP-related ones. Constraint (11) ensures the standard flow conservation rules. Constraint (12) imposes that for each kth subpath it should pass through exact one node $i \in \mathcal{T}^{c,k}$. Constraint (13) (resp. (14)) means that one flow unit is generated at the origin o_c (resp. terminated at the destination d_c).

In [16], the authors pointed out that the constraint matrix of the ILP-based Lagrangian problem of the constrained SPTP is *totally unimodular* [23], which can guarantee that the relaxed LP has feasible solution(s) at extreme point(s) of an integral polyhedron, i.e., the solution(s) of the original ILP. Here, the constraints (9)–(14) of the reformulated Lagrangian problem $\Phi_{\mathrm{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma})$ are the same as those of the Lagrangian problem of the constrained SPTP formulated in [16], i.e., totally unimodular. The difference between our problem $\Phi_{\mathrm{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma})$ and the problem in [16] is the Lagrangian terms of the objective function. Consequently, the linear program $\Phi_{\mathrm{SC}}^{\mathrm{LP}}(\boldsymbol{\mu}, \boldsymbol{\gamma})$ of the Lagrangian problem $\Phi_{\mathrm{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma})$ can reduce the practical computation time while maintaining its optimality.

4.4 Depth First Tour Search

To further reduce computational complexity, we straightforwardly apply the depth first tour search (DFTS) [8], i.e., the shortest path tour finding algorithm, to solve the Lagrangian relaxation $\Phi_{\mathrm{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma})$. For certain $\boldsymbol{\mu}$ and $\boldsymbol{\gamma}$, $\Phi_{\mathrm{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma})$ belongs to an SPTP. Since the objective function $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\gamma})$ is the total path cost (delay) minus a constant $\left(\sum_{(i,j)\in\mathcal{E}} \gamma_{i,j} B_{i,j} + \sum_{i\in\mathcal{V}} \mu_i P_i\right)$, we can redefine the link cost $d_{i,j}$ as follows:

$$d_{i,j} = \begin{cases} d_i^{\mathrm{node}} + d_{i,j}^{\mathrm{link}} + \mu_i p_c^{\mathrm{node}} + \gamma_{i,j} b_c, & \text{if } (i,j) \in \mathcal{E}, \\ d_{i,j}^{\mathrm{func}} + \mu_i p_{c,f}^{\mathrm{func}}, & \text{if } (i,j) \in \hat{\mathcal{E}}^{\mathrm{out}}, \\ 0, & \text{otherwise.} \end{cases}$$

To efficiently solve the SPTP, we apply the DFTS, which first calculates the minimum cost for each subpath from $k = 1$ to $k = K_c + 1$ and then applies the Dijkstra algorithm to obtain the shortest path. To apply this algorithm, we also use the augmented network illustrated in Fig. 3 to guarantee the connectivity between kth and $(k+1)$th subpaths $(k \in \mathcal{K}_c)$.

4.5 Subgradient Algorithm

The last task is solving the Lagrangian dual problem, i.e., finding appropriate $\boldsymbol{\mu}$ and $\boldsymbol{\gamma}$ to maximize $\Phi_{\mathrm{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma})$. We use the subgradient algorithm [23] to solve it. The algorithm first initializes parameters (i.e., τ, $\boldsymbol{\mu}$, and $\boldsymbol{\gamma}$) (line 1) and then solves the Lagrangian problem $\Phi_{\mathrm{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma})$ using $\textsc{solve}(\Phi_{\mathrm{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma}))$ function, which corresponds to either an existing solver, e.g., CPLEX, to solve the relaxed LP $\Phi^{\mathrm{LP}}_{\mathrm{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma})$ or DFTS to solve $\Phi_{\mathrm{SC}}(\boldsymbol{\mu}, \boldsymbol{\gamma})$ (line 3). If the algorithm fails in solving the problem, it stops and no solution is found (lines 4 and 6). Otherwise, it derives the *subgradient* $\boldsymbol{h}^{\tau}$ (resp. $\boldsymbol{g}^{\tau}$) of the corresponding variable $\boldsymbol{\mu}$ (resp. $\boldsymbol{\gamma}$) by calling $\textsc{subgradient}(\Phi_{\mathrm{SC}}(\boldsymbol{\mu}^{\tau}, \boldsymbol{\gamma}^{\tau}))$ function (line 7). The algorithm udpates $\boldsymbol{h}^{\tau}$ and $\boldsymbol{g}^{\tau}$ by the following rules:

$$\begin{aligned}
h_i^{\tau} &= \sum_{c\in\mathcal{C}} \big(p_c^{\mathrm{node}} \sum_{k\in\mathcal{K}_c^{+}} \sum_{(i,j)\in\mathcal{E}} x_{i,j}^{c,k,\tau} + p_{c,f}^{\mathrm{func}} \sum_{k\in\mathcal{K}_c} \sum_{(j,i)\in\hat{\mathcal{E}}^{\mathrm{out}}} x_{j,i}^{c,k,\tau}\big) - P_i,\\
g_{i,j}^{\tau} &= \sum_{c\in\mathcal{C}} b_c \sum_{k\in\mathcal{K}_c^{+}} x_{i,j}^{c,k,\tau} - B_{i,j},
\end{aligned}$$

where h_i^{τ} (resp. $g_{i,j}^{\tau}$) denotes ith element of $\boldsymbol{h}^{\tau}$ (resp. (i,j)th element of $\boldsymbol{g}^{\tau}$). Based on the subgradient and step size at iteration τ (i.e., $\theta_{\boldsymbol{\mu}}^{\tau}$ and $\theta_{\boldsymbol{\gamma}}^{\tau}$), it updates $\boldsymbol{\mu}^{\tau+1}$ and $\boldsymbol{\gamma}^{\tau+1}$ (lines 9–10):

$$\begin{aligned}
\mu_i^{\tau+1} &= \max\{0, \mu_i^{\tau} + \theta_{\boldsymbol{\mu}}^{\tau} h_i^{\tau}\}, &(15)\\
\gamma_{i,j}^{\tau+1} &= \max\{0, \gamma_{i,j}^{\tau} + \theta_{\boldsymbol{\gamma}}^{\tau} g_{i,j}^{\tau}\}, &(16)\\
\theta_{\boldsymbol{\mu}}^{\tau} &= \omega/(\sqrt{\tau}||\boldsymbol{h}^{\tau}||), \quad \theta_{\boldsymbol{\gamma}}^{\tau} = \omega/(\sqrt{\tau}||\boldsymbol{g}^{\tau}||),
\end{aligned}$$

where ω denotes a weighting parameter. If h_i^{τ} (resp. $g_{i,j}^{\tau}$) is less than or equal to zero, (7) (resp (6)) holds, and thus the algorithm will reduce $\mu_i^{\tau+1}$ (resp. $\gamma_{i,j}^{\tau+1}$) as in (15) (resp. (16)) to weaken the penalty for violating (7) (resp. (6)). Otherwise, it strengthen the penalty for violating (7) (resp. (6)) by increasing $\mu_i^{\tau+1}$ (resp. $\gamma_{i,j}^{\tau+1}$) as in (15) (resp. (16)).

The algorithm repeats this procedure by gently decreasing the step size (line 8) until satisfying the stop conditions given by $\textsc{stop\_condition}()$ function (lines 2–12). To be more precise, it will stop when holding one of the following three conditions. The first condition is the optimal stop condition where the solution $\boldsymbol{x}^*$ of the Lagrangian problem $\Phi_{\mathrm{SC}}(\boldsymbol{\mu}^{\tau}, \boldsymbol{\gamma}^{\tau})$ also becomes that of the original CSPTP-based ILP $\mathrm{Z}_{\mathrm{SC}}(\boldsymbol{x})$ by satisfying the original CSPTP-related constraints at the first iteration. The second condition is the relative-gap based stop condition where $\boldsymbol{x}^*$ satisfies the CSPTP-related constraints and the relative improvement ratio of the objective value between τth and $(\tau+1)$th iteration, i.e., $(\mathcal{L}^{\tau+1} - \mathcal{L}^{\tau})/\mathcal{L}^{\tau}$, is less than or equal to the optimality tolerance ε. The last condition is the iteration-limit based condition where the number of iterations reaches a predefined threshold $T_{\max}$.

Table 2 Network scale comparison among six problems.

Network model	# of nodes	# of links
Augmented network in $\mathrm{Z}_{\mathrm{SC}}(\boldsymbol{x})$	$V+\lvert\bigcup_{c\in\mathcal{C}}\mathcal{R}_c\rvert$	$E+2\lvert\bigcup_{c\in\mathcal{C}}\mathcal{R}_c\rvert N$
Augmented network in $\mathrm{Z}_{\mathrm{SCFP}}(\boldsymbol{x})$	$V+F$	$E+2FN$
Augmented network in $\Phi_{\mathrm{SC}}(\boldsymbol{x})$	$V+\lvert\bigcup_{c\in\mathcal{C}}\mathcal{R}_c\rvert N$	$E+2\lvert\bigcup_{c\in\mathcal{C}}\mathcal{R}_c\rvert N$
Augmented network in $\Phi_{\mathrm{SCFP}}(\boldsymbol{x})$	$V+FN$	$E+2FN$
Layered graph model [18, 20]	$\sum_{c\in\mathcal{C}}(K_c+1)V$	$\sum_{c\in\mathcal{C}}(K_c+1)E+K_cN$
Expanded network model [21]	$\sum_{c\in\mathcal{C}}(K_c+1)V$	$\sum_{c\in\mathcal{C}}(K_c+1)E+K_cN$

4.6 Applicability to Service Chaining and Function Placement

The above-mentioned Lagrangian heuristics for SC can easily be extended to that for SCFP by replacing the network model of $\mathrm{Z}_{\mathrm{SC}}(\boldsymbol{x})$ with that of $\mathrm{Z}_{\mathrm{SCFP}}(\boldsymbol{x})$. More specifically, we formulate the Lagrangian problem $\Phi_{\mathrm{SCFP}}(\boldsymbol{x})$ of the CSPTP-based ILP $\mathrm{Z}_{\mathrm{SCFP}}(\boldsymbol{x})$ for SCFP, modify Algorithm 1 by replacing $\Phi_{\mathrm{SC}}(\boldsymbol{x})$ with $\Phi_{\mathrm{SCFP}}(\boldsymbol{x})$, and finally solve the Lagrangian dual problem in the same manner.

4.7 Computational Complexity

Finally, we discuss the computational complexity of the DFTS-based Lagrangian heuristics. (Note that the computational complexity of the LP-based Lagrangian heuristics is the complexity class **P**.) The Lagrangian heuristics serves C SCRs and iterates the while loop (lines 2–12) at most $T_{\max}$ times. In each loop, DFTS, i.e., $\textsc{solve}(\Phi_{\mathrm{SC}}(\boldsymbol{\mu},\boldsymbol{\gamma}))$, in line 3 becomes bottleneck. The computational complexity of DFTS is given by $\mathcal{O}(C(K_c+1)V^+)+\mathcal{O}(C(K_c+1)E^+\log V^+)$, where the first term means the computational complexity of the calculation of the minimum cost for each subpath $k\in\mathcal{K}_c^+$ for the C SCRs and the second one means the computational complexity of the Dijkstra algorithm to obtain the entire shortest path for each of the C SCRs [8]. As a result, the computational complexity of the Lagrangian heuristics based on DFTS becomes $\mathcal{O}(T_{\max}C(K_c+1)V^+)+\mathcal{O}(T_{\max}C(K_c+1)E^+\log V^+)$.

In case of the batch processing (i.e., $C\geq 2$), we confirm that the complexity of DFTS to solve $\Phi_{\mathrm{SC}}(\boldsymbol{\mu},\boldsymbol{\gamma})$ and $\Phi_{\mathrm{SCFP}}(\boldsymbol{\mu},\boldsymbol{\gamma})$ increases with the size of C. To further reduce the computation time, we can adopt parallel computation to DFTS to derive the shortest path tour for each of C SCRs, i.e., $\textsc{solve}(\Phi_{\mathrm{SC}}(\boldsymbol{\mu},\boldsymbol{\gamma}))$ and $\textsc{solve}(\Phi_{\mathrm{SCFP}}(\boldsymbol{\mu},\boldsymbol{\gamma}))$. DFTS can simultaneously calculate the shortest path tour for each of C SCRs under the capacity constraints by sharing the penalty to the capacity violation of physical nodes and links among them. Note that in terms of the ILP and LP, the existing solver (i.e., CPLEX) also supports the parallel optimization.

Table 2 presents the network scale comparison among the augmented networks used in the above mentioned four problems and the two kinds of existing network models, i.e., layered graph model [18, 20] and expanded network model [21]. For simplicity, we assume that $N_f=N$ $(N>0, f\in\mathcal{F})$. We first

Table 3 Service chain demand and requirements [20, 32] of 5 aggregated users (NAT: Network Address Translator, FW: Firewall, TM: Traffic Monitor, WOC: WAN Optimization Controller, IDPS: Intrusion Detection Prevention System, and VOC: Video Optimization Controller).

Service	Sequence of functions	Demand	b_c
Web service	NAT-FW-TM-WOC-IDPS	18.2%	500 kbps
VoIP	NAT-FW-TM-FW-NAT	11.8%	320 kbps
Video streaming	NAT-FW-TM-VOC-IDPS	69.9%	20 Mbps
Online gaming	NAT-FW-VOC-WOC-IDPS	0.1%	20 Mbps

Table 4 Relationship between function type, processing requirements per SCR (5 aggregated users), and demand [32] ($p_c^{\text{node}} = 0.0025$).

Function type	$p_{c,f,k}^{\text{func}}$	Demand
NAT	0.0046	22.6%
FW	0.0045	22.6%
TM	0.0665	20.2%
IDPS	0.0535	16.6%
VOC	0.0270	14.2%
WOC	0.0270	3.7%

focus on the difference between the Lagrangian problems and ILPs. The number of imaginary nodes in the Lagrangian problems increases compared with that in the ILPs, due to the network transformation to guarantee the connectivity between kth and $(k+1)$th subpaths. The impact of this increase on the system performance will be evaluated in Section 5. Focusing on the difference between SC and SCFP, we confirm that the size of augmented network for SC (resp. SCFP) increases in proportion to the product of $|\bigcup_{c\in\mathcal{C}}\mathcal{R}_c|$ and N (resp. that of F and N). Note that $|\bigcup_{c\in\mathcal{C}}\mathcal{R}_c|$ is the number of functions required by C SCRs while F is the total number of functions (i.e., $|\bigcup_{c\in\mathcal{C}}\mathcal{R}_c| \leq F$).

Next, focusing on the scale difference between the augmented network models and the existing network models, we can confirm that the augmented network model, i.e., $\mathrm{Z}_{\mathrm{SC}}(\boldsymbol{x})$ and $\Phi_{\mathrm{SC}}(\boldsymbol{x})$, becomes smaller than layered graph and expanded network models with increase of K_c. In addition, these existing network models require to construct a tailor-made network adaptive to the arrival of new SCR(s) while our approach can reuse an augmented network for arbitrary SCRs.

5 Numerical Results of Online Service Chaining

In this section, we evaluate the effectiveness of the proposed heuristics in terms of the computational complexity and performance of SC. For the evaluation, the server with Intel Core i9-9900K 8 core and 64 GB memory is used.

5.1 Evaluation Scenario

To evaluate the proposed heuristics in terms of the scalability and optimality, we consider a carrier network based on a physical network with 200 physical nodes, inspired by [20, 21, 24]. Physical links are randomly generated between two arbitrary physical nodes with the probability $\pi = 0.032$ as in [33]. As in [20, 21, 24], we expect that if the proposed heuristics works well in the above network, it will also work in other networks with simpler structures, e.g., fat-trees [34]. We assume that each physical node i has the processing resources of 10-core CPU, i.e., $P_i = 10$, and each physical link between physical nodes i and j $(i, j \in \mathcal{V}, i \neq j)$ has identical bandwidth capacity $B_{i,j} = 10\,\text{Gbps}$. Each physical link delay $d^{\text{link}}_{i,j}$ between two physical nodes i and j follows a uniform distribution in the range of $[9,\ 11]\,\text{ms}$ and takes the average of $10\,\text{ms}$. The traversal delay d^{node}_v at physical node $v \in \mathcal{V}$ (resp. the processing delay $d^{\text{func}}_{\hat{v}_f,v}$ for executing function $f \in \mathcal{F}$ at physical node $v \in \mathcal{N}_f$) follows a uniform distribution in the range of $[0.09,\ 0.11]\,\text{ms}$ (resp. $[45,\ 55]\,\text{ms}$) and takes the average of $0.1\,\text{ms}$ (resp. $50\,\text{ms}$).

We use the service chain demand and requirements in Table 3. We assume six function types ($F = 6$) and four service types, each of which consists of five functions ($K_c = 5$). In this SC scenario, we randomly choose 30 physical nodes as VNF-enabled ones ($V_{\text{VNF}} = 30$) and further allocate N_f VNF-enabled physical nodes randomly chosen from them to each function $f \in \mathcal{F}$. Note that in the SCFP scenario, which will be given in Section 6, we will reveal the optimal number and locations of functions and the impact of the number of VNF-enabled physical nodes. For each SCR c, we select one of the services according to the demand distribution in Table 3. Every SCR c serves 5 aggregated users, which requires the aggregate processing requirement for data forwarding, $p^{\text{node}}_c = 0.0025$, and that for each function $f \in \mathcal{F}$, $p^{\text{func}}_{c,f,k}$, given in Table 4. Note that Table 4 also gives the demand for each function $f \in \mathcal{F}$ as a result of the service selection according to Table 3. We randomly choose the origin node o_c and destination node d_c of SCR c from the set $\mathcal{V}$ of physical nodes such that $o_c \neq d_c$.

We consider the online SC (i.e., $C = 1$) to focus on network services with high real-time property. A simulator for the online SC was implemented in C++ programming language with the boost graph library [35]. In the simulation, we assume that the simulator obeys the following queuing model. A new SCR c arrives at the system according to a Poisson process with parameter $\lambda > 0$. The new request will be added to the end of the queue with infinite buffer. The NFV orchestrator tries to calcualte an appropriate service path for each SCR in a first come first served (FCFS) manner. If the NFV orchestrator succeeds in finding the service path that meets both the resource constraints and service chain requirements, it will establish the corresponding service path with the service delay. Considering the fact that services in Table 3 tend to require long-term communications, we assume that each established service path persistently occupies the allocated resources of physical nodes and links during the simulation. Through preliminary experiments, we confirmed that the NFV network capacity, i.e., the number of SCRs that the NFV network can accommodate, becomes 713 in average if the optimal resource allocation is realized, which can be regarded as the solution of the CSPTP-based ILP with no execution time. We set the simulation time to be 100 s such that the NFV network eventually reaches this saturated situation by increasing λ.

We evaluate two types of the proposed Lagrangian heuristics for SC, i.e., *LP-based Lagrangian heuristics* and *DFTS-based Lagrangian heuristics*. Through the preliminary experiments, we use $\delta = 0.05$, $\omega = 100$, and $T_{\max} = 3$. For comparison purpose, we also evaluate the *CSPTP-based ILP* and the *greedy algorithm* [10]. The CSPTP-based ILP can be regarded as the optimal comparison scheme in terms of the total delay (i.e., objective function) because it can always achieve the minimum total delay if the execution time can be ignored. On the other hand, the greedy algorithm can be regarded as the comparison scheme with the minimum computational complexity. The greedy algorithm decomposes the service path into $K_c + 1$ subpaths and sequentially tries to calculates a shortest path for each subpath from the beginning ($k = 1$) to the end ($k = K_c + 1$). As a result, its computational complexity becomes $\mathcal{O}(C(K_c{+}1)E^+ \log V^+)$ [10]. As for the proposed heuristics and CSPTP-based ILP, we can apply the parallel computation architectures. In case of the DFTS-based Lagrangian heuristics, we use OpenMPI [36] to calculate the shortest path tour per SCR in parallel. For the CSPTP-based ILP and LP-based heuristics, we use the existing solver CPLEX 12.8 to solve them where CPLEX supports the parallel optimization and the number of threads is set to be 32.

We should note here that each scheme will require the execution time depending on not only its computational complexity but also both the service chain requirements and remaining network capacity. In this paper, we consider two cases in terms of the execution time: *ideal case* and *realistic case*. The ideal case will be used to reveal the performance limit of each scheme under no execution time. As for evaluation criteria, we define the *acceptance ratio* as the ratio of the number of SCRs served successfully to the total number of SCRs. Note that the remaining SCRs in the queue at the simulation end will be rejected. We evaluate the computational complexity from the viewpoint of the *average execution time*, which is the average amount of time spent finding a service path per SCR. In context of queueing theory, we can interpret the execution time as the service time. To measure how busy the system is, we evaluate the *traffic intensity* $\rho = \lambda/\mu$ where μ denotes the average service rate. We also evaluate the objective function, i.e., *average total delay of service paths* among all accepted requests. In what follows, we show the simulation results in the average of 50 independent simulation runs.

5.2 Optimality of Service Chaining

In this section, we first focus on the maximum performance of resource allocation under the ideal case where the execution time is ignored. Fig. 4 depicts the relationship between the arrival rate λ and total delay. We confirm that the total delay worsens with the arrival rate λ, regardless of the schemes. This is because the increase of λ leads to more resource consumption, which may make the new SCR to establish a longer service path due to the capacity constraint. We, however, observe that both the proposed heuristics approaches, i.e., the

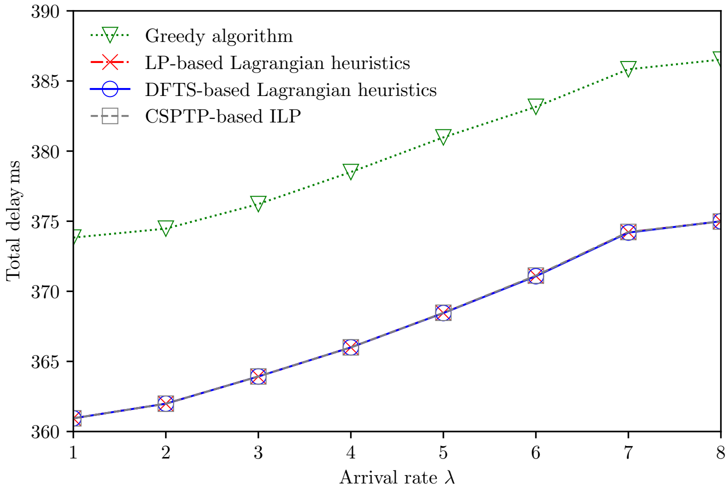

Fig. 4 Impact of arrival rate λ on total delay (SC under ideal case).

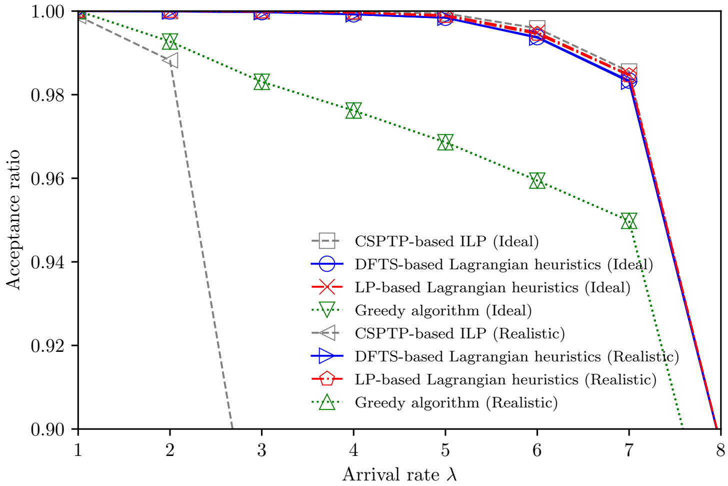

Fig. 5 Impact of arrival rate λ on acceptance ratio (SC under ideal case and realistic case).

LP-based Lagrangian heuristics and DFTS-based Lagrangian one, exhibit the total delay competitive with the CSPTP-based ILP and reduce the total delay by 11.5–12.9 ms compared with the greedy algorithm, respectively.

Fig. 5 illustrates the relationship between the arrival rate λ and the acceptance ratio in the ideal case. Note that we will discuss the results in the realistic case in Section 5.3. Simply speaking, resource allocation with low network utilization will contribute to high acceptance ratio. In our problem, the objective function is the minimization of total delay of service paths, which will also reduce the network utilization by making the path with a less number of hop count (intermediate nodes and links). Focusing on the results in

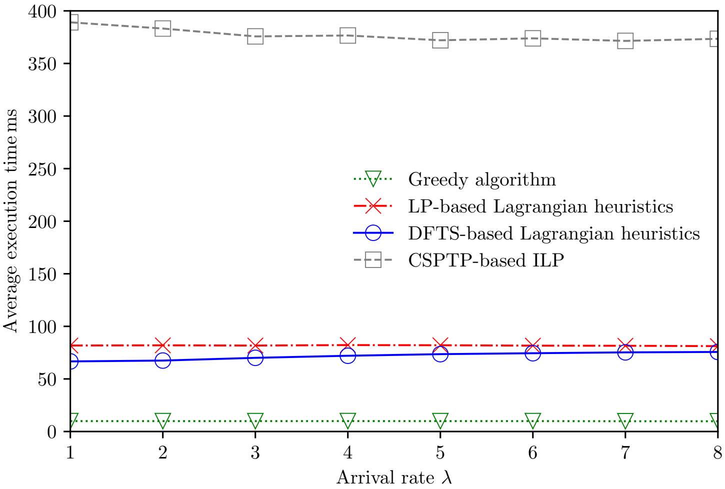

Fig. 6 Impact of arrival rate λ on average execution time (SC under realistic case).

the ideal case, we observe that the four schemes have similar tendency: 1) The acceptance ratio maintains almost one in the range of $\lambda \in [1, \hat{\lambda}]$ and then, 2) it steeply decreases with increase of λ. However, the value of $\hat{\lambda}$ differs among the schemes. The algorithm has $\hat{\lambda} = 2$ and exhibits the worst performance while the two proposed heuristics can achieve almost the same optimal performance as the CSPTP-based ILP. Specifically, the performance degradation of the two proposed heuristics, i.e., LP-based Lagrangian heuristics and DFTS-based Lagrangian one, is suppressed by 0.22% and 0.08% compared with that of the CSPTP-based ILP, respectively.

5.3 Tradeoff between Optimality of Service Chaining and Computational Complexity

Next, we consider the realistic case where the execution time is not negligible and prone to change according to the schemes. Fig. 6 depicts how the arrival rate λ affects the average execution time in the realistic case. Since the average execution time is the average amount of time spent finding a service path per SCR, it is almost constant regardless of the value of λ for all schemes. Note that the total amount of execution time for serving all the arrived SCRs can approximately be estimated as the product of λ and average execution time, which linearly increases with λ. We also observe that the two proposed heuristics and greedy algorithm are much faster than the CSPTP-based ILP.

We should note here that the performance degradation will be caused by the following two factors, i.e., resource depletion or long execution time. In Section 5.2, we only focused on the impact of resource depletion through evaluations under the ideal case. In what follows, we focus on the performance degradation caused by the long execution time. Fig. 5 illustrates impact of the arrival rate λ on the acceptance ratio in the realistic case, in addition to

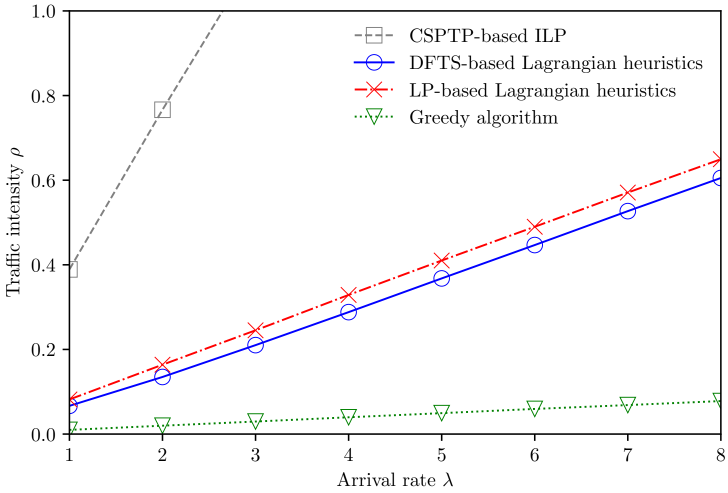

Fig. 7 Relationship between the arrival rate λ and traffic intensity ρ (SC under realistic case).

that in the ideal case. Different from the result in the ideal case, we observe that the acceptance ratio of the CSPTP-based ILP in the realistic case drastically decreases when $\lambda \geq 3$. This is caused by the heavily loaded state of the system where the traffic intensity ρ becomes larger than one, which can be confirmed through the relationship between the arrival rate λ and traffic intensity ρ, as shown in Fig. 7. On the contrary, we find that the greedy algorithm shows almost the same performance in both the ideal and realistic cases, which implies that the performance degradation of greedy algorithm comes from the resource depletion. As for the two proposed heuristics in the realistic case, we confirm that they can perform well as in the ideal case by achieving efficient resource allocation in a speedy manner.

6 Numerical Results of Service Chaining and Function Placement

In this section, we evaluate the solution optimality and computational complexity of the proposed heuristics in terms of SCFP. In the calculation, we used the same server used in Section 5.

6.1 Evaluation Scenario

We use the same simulator and settings in Section 5.1, except the following. We consider the batch processing with $C = 10$ to evaluate the performance of SCFP. Every 10 SCR arrivals, each of which follows a Poisson process with $\lambda = 8$, the NFV orchestrator tries to find the optimal service paths for the C SCRs and determine the number and locations of functions. As for the average

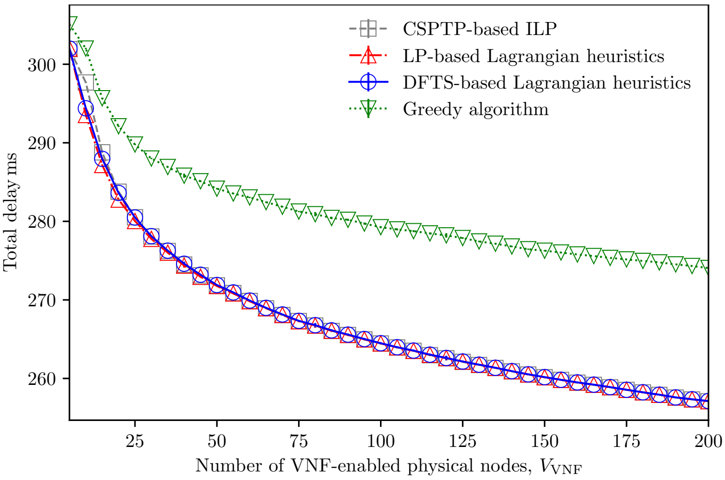

Fig. 8 Impact of the number V_{VNF} of VNF-enabled physical nodes on total delay ($C = 10$, $\lambda = 8$, SCFP case).

execution time, we define it as the average amount of time spent finding service paths per SCRs with batch size of $C = 10$.

In this evaluation, we assume that VNF-enabled physical nodes and legacy ones can coexist, which will happen during the transition period from the legacy networks to NFV networks. We assume that each VNF-enabled physical node can execute any function $f \in \mathcal{F}$ under the capacity constraint. We change the number V_{VNF} of VNF-enabled physical nodes in the range of $[5, V]$ and randomly select V_{VNF} VNF-enabled physical nodes from all physical nodes. Please note that the actual number N_f^* and location(s) of VNF-enabled physical nodes possessing function $f \in \mathcal{F}$ will be determined by solving the SCFP problem.

As mentioned in Section 3.5, the CSPTP-based ILP for SC and greedy algorithm for SC can be extended to solve the SCFP by modifying the augmented network. Similarly, the proposed heuristics can also be extended to those for SCFP by modifying the augmented network as mentioned in Section 4.3.

6.2 Tradeoff between Optimality of Service Chaining and Function Placement and Computational Complexity

Intuitively, the optimality of SCFP will improve with increase of the number V_{VNF} of VNF-enabled physical nodes at the cost of computational complexity. In this section, we will figure out this tradeoff. Fig 8 illustrates the relationship between V_{VNF} and total delay for the four schemes. As we expected, the increase of V_{VNF} leads to the reduction of the total delay, regardless of the schemes. This is because the combination of service chaining and function location can determine both the appropriate number and locations of functions to minimize the total delay of service paths. Specifically, the total delay drastically decreases even under small V_{VNF} (e.g., $V_{\mathrm{VNF}} = 30$) in case

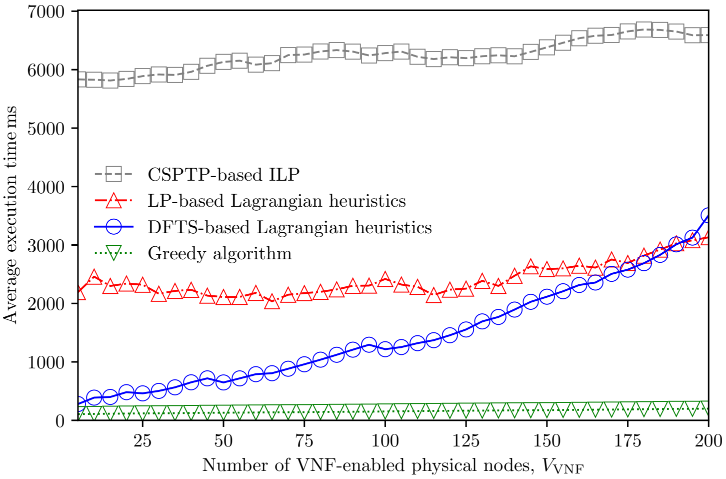

Fig. 9 Impact of the number V_{VNF} of VNF-enabled physical nodes on average execution time ($C = 10$, $\lambda = 8$, SCFP case).

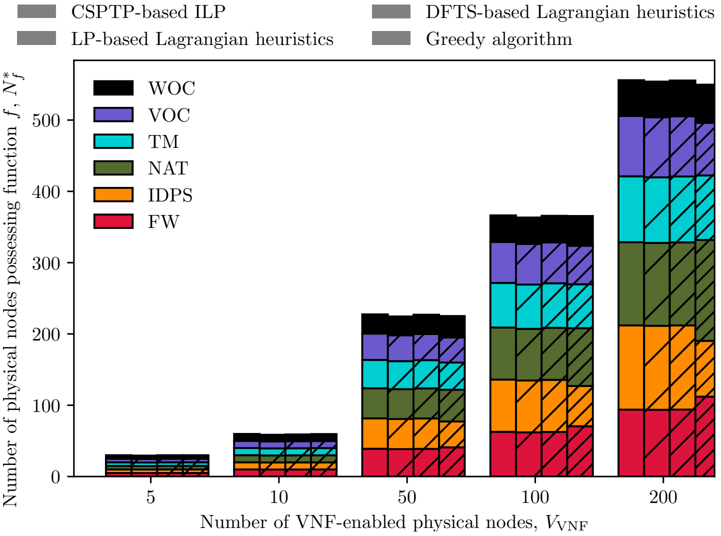

Fig. 10 Impact of the number V_{VNF} of VNF-enabled physical nodes on the number N_f^* of physical nodes possessing function f ($C = 10$, $\lambda = 8$, SCFP case).

of the proposed heuristics (i.e., the DFTS-based Lagrangian heuristics and LP-based one), which indicates that the combination of service chaining and function placement works well even at the early stage of transition to NFV networks. Focusing on the performance difference among the four schemes, we observe that the DFTS-based Lagrangian heuristics and LP-based one show the competitive total delay with the CSPTP-based ILP and reduce the total

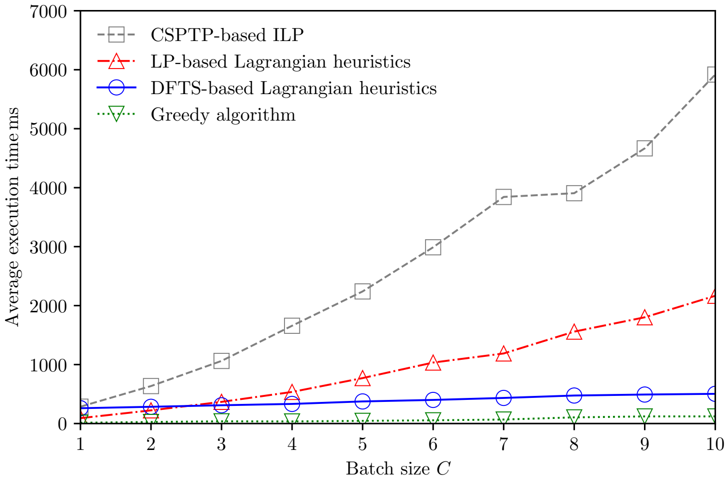

Fig. 11 Impact of batch size C on average execution time ($V_{\mathrm{VNF}} = 30$, $\lambda = 8$, SCFP case).

delay by 3.09–16.96 ms and 3.09–16.97 ms compared with the greedy algorithm, respectively.

Next, Fig. 9 depicts how the number V_{VNF} of VNF-enabled physical nodes affects the average execution time. As we expected, the increase of V_{VNF} leads to the increase of the execution time in all schemes. (The average execution time of the greedy algorithm also shows the slight increase with respect to V_{VNF}.) The proposed heuristics and greedy algorithm achieve smaller average execution time than the CSPTP-based ILP. Specifically, the DFTS-based Lagrangian heuristics (resp. the LP-based Lagrangian heuristics) can reduce the average execution time by 46.8% (resp. 52.5%) compared with the CSPTP-based ILP even in case of $V_{\mathrm{VNF}} = 200$ while keeping the solution optimality, as shown in Fig. 8. The greedy algorithm shows the smallest average execution time at the sacrifice of solution optimality.

Finally, Fig. 10 illustrates how the four schemes adjust the number N_f^* of physical nodes possessing function f ($f \in \mathcal{F}$) by changing the number V_{VNF} of VNF-enabled physical nodes. We first confirm that the four schemes can supply each function according to the corresponding demand given in Table 4, regardless of V_{VNF}. We also observe that N_f^* increases with V_{VNF}. This is because the total delay (i.e., objective function) can be reduced by allocating more instances (copies) of the same function to more VNF-enabled physical nodes, as shown in Fig. 8.

6.3 Impact of Batch Size on Average Execution Time of Service Chaining and Function Placement

As mentioned in Section 3.6, the combination of SCFP is suitable for batch processing because it can achieve more effective resource allocation at the cost of longer execution time, compared with SC. Fig. 11 depicts the impact of batch size C on average execution time. We observe that the average execution time increases with increase of C, regardless of the schemes.

We observe that the DFTS-based (resp. LP-based) Lagrangian heuristics can reduce the average execution time up to 91.5% (resp. 69.0%) compared with the CSPTP-based ILP. The DFTS-based Lagrangian heuristics shows longer (resp. smaller) average execution time than the the LP-based one when $1 \leq C < 3$ (resp. $3 \leq C \leq 10$). This comes from the difference of the parallel computing architectures between them. In case of the LP-based Lagrangian heuristics, CPLEX is used as the solver, which can search for optimal service paths for all SCRs with benefit of parallel computing. On the contrary, the DFTS-based Lagrangian heuristics adopts OpenMPI to calculate the optimal service path per SCR in parallel, which indicates that the benefit of parallel computing only arises when C becomes large.

7 Conclusion

In this paper, we have proposed the two types of the Lagrangian heuristics for the speedy and efficient service chaining (SC), i.e., the linear programming (LP) based one and the depth first tour search (DFTS) based one, by integrating the several existing techniques, i.e., Lagrangian relaxation, linear relaxation, shortest path tour algorithm called DFTS, and subgradient algorithm. In terms of the LP-based Lagrangian heuristics, we have proved that the Lagrangian problem of the original capacitated shortest path tour problem (CSPTP)-based ILP has a totally unimodular constraint matrix, which guarantees the integrality of the decision variables even under the linear relaxation. Furthermore, we have shown that the proposed heuristics can also solve both the service chaining and function placement (SCFP) by extending the augmented network.

Through the numerical results of the online SC, we have shown that the proposed heuristics can perform almost the optimal resource allocation with much smaller execution time, compared with the combination of the CSPTP-based ILP and the existing solver, i.e., CPLEX. In the evaluations of SCFP, we have demonstrated that the proposed heuristics can still balance the solution optimality and computational complexity, thanks to the parallel computation architectures.

Declarantions

Author Contribution

Hara and Sasabe made contributions to the conception and design of the work. Hara implemented the simulator for the experiments and conducted the evaluations. As for the manuscript, Hara prepared all the figures and both Hara and Sasabe contributed to writing and reviewing the whole part of the manuscript. All authors approved the version to be published.

Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

Funding

This work was supported in part by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant 22H03586, 19K11942, and 21K21288, Japan.

References

[1] Hara, T., Sasabe, M.: Lagrangian Heuristics for Capacitated Shortest Path Tour Problem Based Online Service Chaining. In: Proc. of 2022 IEEE/IFIP Network Operations and Management Symposium, pp. 1–9 (2022)

[2] Han, B., Gopalakrishnan, V., Ji, L., Lee, S.: Network Function Virtualization: Challenges and Opportunities for Innovations. IEEE Communications Magazine **53**(2), 90–97 (2015). https://doi.org/10.1109/MCOM.2015.7045396

[3] Bhamare, D., Jain, R., Samaka, M., Erbad, A.: A Survey on Service Function Chaining. Journal of Network and Computer Applications **75**, 138–155 (2016). https://doi.org/10.1016/j.jnca.2016.09.001

[4] Yi, B., Wang, X., Li, K., k. Das, S., Huang, M.: A Comprehensive Survey of Network Function Virtualization. Computer Networks **133**, 212–262 (2018). https://doi.org/10.1016/j.comnet.2018.01.021

[5] Herrera, J.G., Botero, J.F.: Resource Allocation in NFV: A Comprehensive Survey. IEEE Transactions on Network and Service Management **13**(3), 518–532 (2016). https://doi.org/10.1109/TNSM.2016.2598420

[6] Halpern, J., Pignataro, C.: Service Function Chaining (SFC) Architecture. Technical Report RFC7665 (October 2015). https://doi.org/10.17487/RFC7665

[7] Awerbuch, B., Azar, Y., Epstein, A.: The Price of Routing Unsplittable Flow. In: Proc. of STOC. ACM,, vol. 42, pp. 160–177 (2005)

[8] Bhat, S., Rouskas, G.N.: Service-Concatenation Routing with Applications to Network Functions Virtualization. In: Proc. of 26th International Conference on Computer Communication and Networks (ICCCN), pp. 1–9 (2017)

[9] Gao, L., Rouskas, G.N.: Congestion Minimization for Service Chain Routing Problems With Path Length Considerations. IEEE/ACM Transactions on Networking, 1–14 (2020). https://doi.org/10.1109/TNET.2020.3017792

[10] Sasabe, M., Hara, T.: Capacitated Shortest Path Tour Problem Based Integer Linear Programming for Service Chaining and Function Placement in NFV Networks. IEEE Transactions on Network and Service Management **18**(1), 104–117 (2021). https://doi.org/10.1109/TNSM.2020.3044329

[11] Festa, P.: Complexity Analysis and Optimization of the Shortest Path Tour Problem. Optimization Letters **6**(1), 163–175 (2012)

[12] Festa, P., Guerriero, F., Laganà, D., Musmanno, R.: Solving the Shortest Path Tour Problem. European Journal of Operational Research **230**(3), 464–474 (2013)

[13] Festa, P.: The Shortest Path Tour Problem: Problem Definition, Modeling, and Optimization. In: Proc. of INOC, pp. 1–7 (2009)

[14] Ferone, D., Festa, P., Guerriero, F., Laganà, D.: The Constrained Shortest Path Tour Problem. Computers & Operations Research **74**, 64–77 (2016). https://doi.org/10.1016/j.cor.2016.04.002

[15] de Andrade, R.C., Saraiva, R.D.: An Integer Linear Programming Model for the Constrained Shortest Path Tour Problem. Electronic Notes in Discrete Mathematics **69**, 141–148 (2018)

[16] Saraiva, R.D., de Andrade, R.C.: Constrained Shortest Path Tour Problem: Models, Valid Inequalities, and Lagrangian Heuristics. International Transactions in Operational Research **28**(1), 222–261 (2021). https://doi.org/10.1111/itor.12782

[17] Sallam, G., Gupta, G.R., Li, B., Ji, B.: Shortest Path and Maximum Flow Problems Under Service Function Chaining Constraints. In: Proc. of IEEE Conference on Computer Communications, pp. 2132–2140 (2018). https://doi.org/10.1109/INFOCOM.2018.8485996

[18] Hyodo, N., Sato, T., Shinkuma, R., Oki, E.: Virtual Network Function Placement for Service Chaining by Relaxing Visit Order and Non-Loop Constraints. IEEE Access **7**, 165399–165410 (2019)

[19] Sun, G., Li, Y., Yu, H., Vasilakos, A.V., Du, X., Guizani, M.: Energy-Efficient and Traffic-Aware Service Function Chaining Orchestration in Multi-Domain Networks. Future Generation Computer Systems **91**, 347–360 (2019). https://doi.org/10.1016/j.future.2018.09.037

[20] Huin, N., Jaumard, B., Giroire, F.: Optimal Network Service Chain Provisioning. IEEE/ACM Transactions on Networking **26**(3), 1320–1333 (2018). https://doi.org/10.1109/TNET.2018.2833815

[21] Nguyen, T., Girard, A., Rosenberg, C., Fdida, S.: Routing via Functions in Virtual Networks: The Curse of Choices. IEEE/ACM Transactions on Networking **27**(3), 1192–1205 (2019). https://doi.org/10.1109/TNET.2019.2912717

[22] ILOG: IBM ILOG CPLEX Optimizer. https://www.ibm.com/products/ilog-cplex-optimization-studio. Accessed 15 Jun. 2022 (2020)

[23] Schrijver, A.: Theory of Linear and Integer Programming, Reprinted edn. Wiley, Chichester (2000)

[24] Bhamare, D., Samaka, M., Erbad, A., Jain, R., Gupta, L., Chan, H.A.: Optimal Virtual Network Function Placement in Multi-Cloud Service Function Chaining Architecture. Computer Communications **102**, 1–16 (2017). https://doi.org/10.1016/j.comcom.2017.02.011

[25] Li, D., Hong, P., Xue, K., Pei, J.: Virtual Network Function Placement and Resource Optimization in NFV and Edge Computing Enabled Networks. Computer Networks **152**, 12–24 (2019). https://doi.org/10.1016/j.comnet.2019.01.036

[26] Dieye, M., Ahvar, S., Sahoo, J., Ahvar, E., Glitho, R., Elbiaze, H., Crespi, N.: CPVNF: Cost-Efficient Proactive VNF Placement and Chaining for Value-Added Services in Content Delivery Networks. IEEE Transactions on Network and Service Management **15**(2), 774–786 (2018). https://doi.org/10.1109/TNSM.2018.2815986

[27] Soualah, O., Mechtri, M., Ghribi, C., Zeghlache, D.: Online and Batch Algorithms for VNFs Placement and Chaining. Computer Networks **158**, 98–113 (2019). https://doi.org/10.1016/j.comnet.2019.01.041

[28] Alleg, A., Ahmed, T., Mosbah, M., Riggio, R., Boutaba, R.: Delay-aware VNF placement and chaining based on a flexible resource allocation approach. In: Proc. of International Conference on Network and Service Management (CNSM), pp. 1–7 (2017). https://doi.org/10.23919/CNSM.2017.8255993

[29] Kiji, N., Sato, T., Shinkuma, R., Oki, E.: Virtual Network Function Placement and Routing Model for Multicast Service Chaining Based on Merging Multiple Service Paths. In: Proc. of 2019 IEEE 20th International Conference on High Performance Switching and Routing (HPSR), pp. 1–6 (2019). https://doi.org/10.1109/HPSR.2019.8807998

[30] Kiji, N., Sato, T., Shinkuma, R., Oki, E.: Virtual Network Function Placement and Routing for Multicast Service Chaining Using Merged Paths. Optical Switching and Networking **36**, 1–10 (2020). https://doi.org/10.1016/j.osn.2020.100554

[31] Papagianni, C., Leivadeas, A., Papavassiliou, S., Maglaris, V., Cervelló-Pastor, C., Monje, Á.: On the Optimal Allocation of Virtual Resources in Cloud Computing Networks. IEEE Transactions on Computers **62**(6), 1060–1071 (2013). https://doi.org/10.1109/TC.2013.31

[32] Savi, M., Tornatore, M., Verticale, G.: Impact of Processing-Resource Sharing on the Placement of Chained Virtual Network Functions. IEEE Transactions on Cloud Computing, 1–14 (2019). https://doi.org/10.1109/TCC.2019.2914387

[33] Batagelj, V., Brandes, U.: Efficient Generation of Large Random Networks. Physical Review E **71**(3), 1–5 (2005). https://doi.org/10.1103/PhysRevE.71.036113

[34] Leiserson, C.E.: Fat-trees: Universal Networks for Hardware-efficient Supercomputing. IEEE Transactions on Computers **C-34**(10), 892–901 (1985). https://doi.org/10.1109/TC.1985.6312192

[35] Boost: Boost Graph Library. https://www.boost.org/. Accessed 15 Jun. 2022 (2020)

[36] Open MPI: Open Source High Performance Computing. https://www.open-mpi.org/. Accessed 15 Jun. 2022 (2020)