Service Chain Provisioning Model Considering Traffic Changes Due to Virtualized Network Functions

Shintaro Ozaki, *Student Member, IEEE,* Takehiro Sato, *Member, IEEE,* and Eiji Oki, *Fellow, IEEE*

***Abstract*—Service chaining provides network services by flexibly configuring service chains that connect virtualized network functions (VNFs) in the appropriate order so as to satisfy users' needs. Existing models can be inefficient in terms of consuming network and computation resources since the models do not consider the traffic changes due to VNFs or the models restrict routing and VNF placement. This paper proposes a service chain provisioning model that handles the traffic changes created by VNFs while determining the VNF visit order of each request, request routes, and VNF placement. The service chain provisioning problem is formulated as an integer linear programming (ILP) problem. Three methods for limiting the number of VNF visit order patterns considered in the ILP problem are introduced to shorten the computation time. In order to handle a problem that is intractable with the ILP model, we introduce a greedy algorithm and an algorithm that divides the problem into the VNF placement part and the routing part. Numerical results show that considering the traffic changes due to VNFs yields more efficient consumption of network and computation resources than the alternative of assuming that the traffic amount of each request is constant between the endpoints. The results also show that the computation time can be shortened in our examined scenarios while we obtain the objective value larger by at most 0.4% than the optimal value by limiting the number of visit order patterns considered.**

***Index Terms*—network function virtualization, service chaining, traffic changing effect, integer linear programming.**

I. Introduction

Network function virtualization (NFV) is a technology that realizes network functions such as firewalls, wide area network (WAN) optimization, and intrusion detection systems (IDSs) as software-based virtualized network functions (VNFs) running on commercial off-the-shelf (COTS) servers [1], [2]. Traditionally, dedicated hardware devices are needed in order to install network functions. Such hardware devices need to be installed in particular places and be configured individually; this incurs excessive time and cost overheads. By virtualizing networks, we only need to purchase licenses for the VNFs desired and install them in COTS servers, so we can expect significant cost reductions. Furthermore, we can easily install, uninstall, or update VNFs on demand, so we can provide network services to users more rapidly and flexibly.

Service chaining is a technology that provides the user with the network service needed by linking VNFs in the appropriate order [3]. We call each concatenation of VNFs a service chain. The user makes a request to a network provider for provisioning a service chain. The service chain traverses the network from its source to its destination while passing through required VNFs in the appropriate order. In order to instantiate network services for users, we need to determine the best request routes and VNF placement so as to minimize the total cost of service chain provisioning, while satisfying some constraints of computation resources and transmission capacity.

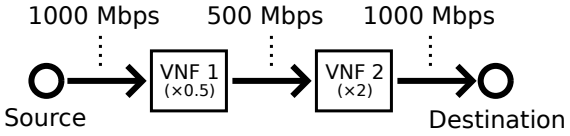


Fig. 1. Example of service chain.

Some VNFs change the traffic amount when processing a request [4], [5]. For example, a Bose-Chaudhuri-Hocquenghem (BCH) encoder increases traffic since it adds checksum information to each packet. On the other hand, WAN optimizers and firewalls decrease the traffic amount since they compress and discard packets, respectively.

There are several studies on service chain provisioning. Huin *et al.* [6] introduced a logical layered network consisting of replicas of the physical network topology in order to relax routing constraints. A logical layered network enables a request route to pass through the same link in the same direction more than once, which offers flexible determination of request routes. Allybokus *et al.* [7] introduced a parameter representing constraints on VNF visit order so as to solve an optimization problem for service chain provisioning while relaxing visit order constraints. Hyodo *et al.* [8] introduced a logical layered network and a decision variable that represents node visit order so as to relax both visit order constraints and routing constraints. However, none of these studies consider the impact of traffic amount changed by VNFs; they assume each request has constant traffic from end to end. This assumption demands that the route be provisioned for the maximum possible traffic value along the entire route. This implies that the transmission capacity reserved on some links can be more than the actual traffic. For example, consider the service chain in Fig. 1. The initial transmission capacity of the request is $1000\,\mathrm{Mbps}$. VNF 1 halves the traffic changing rate to $500\,\mathrm{Mbps}$ while VNF 2 doubles the traffic to $1000\,\mathrm{Mbps}$. In this case, reserving the transmission capacity at least $1000\,\mathrm{Mbps}$ over the entire route wastes transmission capacity on the path between VNF 1 and VNF 2.

Several works dealt with service chain provisioning problems in consideration of traffic changing effects of VNFs [4],

This work was supported in part by JSPS KAKENHI, Japan, under Grant Numbers 19K14980 and 21H03426.

S. Ozaki, T. Sato, and E. Oki are with the Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan (e-mail: sozaki@icn.cce.i.kyoto-u.ac.jp, takehiro.sato@i.kyoto-u.ac.jp, oki@i.kyoto-u.ac.jp.)

[5], [9], [10], which lack flexibility in determining the request routes and the VNF placement. The model in [4] is not applicable to the case where the request routes and the VNF placement are determined for multiple requests simultaneously. Furthermore, the model in [4] does not allow looping routes. The model in [5] determines only the VNF placement; if we use this model for service chaining, we must construct request routes after the VNF placement. The model in [9] assumes that request routes are predetermined; VNFs are deployed on only nodes that are included in the predetermined routes. The model in [10] determines a request route by choosing the best of several candidates; the required VNFs of each request are forced to be placed only on the nodes of the predetermined route candidates. Service chain provisioning with these models can lead to inefficient use of the network and computation resources. In order to reduce the cost of service chain provisioning, we need a model that relaxes the constraints in routing and the VNF placement while considering traffic changing effects of VNFs.

This paper proposes a service chain provisioning model considering VNF traffic changes while determining the VNF visit order of each request, request routes, and VNF placement flexibly. The proposed model determines request routes and VNF placement so as to minimize the total cost, i.e., the sum of the link utilization cost and the VNF placement cost. The proposed model ensures that only the transmission capacity needed to satisfy each request is reserved on a link-by-link basis. The proposed model uses a logical layered network to handle VNF traffic changes and looping routes. The main contributions of this work in conjunction with the proposed model are summarized as follows:

- We formulate the proposed model as an optimization problem. If the transmission capacity of each request on each layer is given as a decision variable and parameters of visit order constraints are introduced in the optimization problem, the objective function and some constraints become non-linear. Therefore, the proposed model is transformed into an integer linear programming (ILP) problem that takes the transmission capacity as a constant by calculating it in advance.
- We introduce three methods to limit the number of visit order patterns to shorten the computation time while keeping the total cost as close to the optimal value as possible. We use the algorithm presented in [4] to determine visit order patterns considered in the ILP problem.
- We introduce two heuristic algorithms for the proposed model. One is a greedy algorithm that iteratively solves the ILP problem with a limited number of requests. The other is an algorithm that divides the problem into the VNF placement part and the routing part.
- We evaluate the performances of the proposed model and heuristic algorithms. Numerical results show that the total cost can be reduced by considering VNF traffic changes and allowing looping routes. The computation time can be shortened by limiting the number of visit order patterns considered. We also observe that the heuristic algorithms obtain request routes and VNF placement in a scenario where it is hard to obtain an optimal solution by solving the ILP problem.

This work is an extended version of [11]. The extensions to [11] are described as follows. We survey existing researches related to our work. We introduce heuristic algorithms for the proposed model. We compare the performances of heuristic algorithms in terms of the computation time and the total cost. We extensively evaluate the performance of proposed model and heuristic algorithms with various order constraints.

The rest of this paper is organized as follows. Section II explains related technologies and algorithms. Section III explains the proposed model and the formulation of the service chain provisioning problem. Section IV introduces the methods used to limit the number of visit order patterns to shorten the computation time of the proposed model. Section V introduces the heuristic algorithms. Section VI shows numerical results. Finally, Section VII concludes this paper.

II. Related works

A. Overview of studies on service chaining

Table I shows the summary of studies on service chaining, which are related to this work in that the studies dealt with optimization problems that determine request routes and VNF placement.

The studies in [6]–[8], [12]–[15] dealt with service chaining problems where the required transmission capacity of each request is constant between the endpoints. Huin *et al.* [6] dealt with a problem that determines the VNF placement and request routes while allowing looping routes by using a logical layered network. The authors presented a column generation based scheme that enables to obtain a near-optimal solution of the problem, which is intractable with an ILP solver when the scale of a given network is large. Allybokus *et al.* [7] dealt with a problem that determines VNF visit orders of requests in addition to the VNF placement and request routes under circumstances in which there are relative orders between VNFs. Hyodo *et al.* [8] presented a model that allows looping routes and relaxes order constraints at the same time. Sasabe *et al.* [12] presented a routing and VNF placement model based on an augmented network in order to allow looping routes. Xu *et al.* [13] showed that setup and operation costs increase when a set of required VNFs is a totally-ordered set compared to when there is no order constraint. The authors presented a dynamic programming based scheme that determines the VNF placement and request routes in order to deal with requests arriving at or leaving from a network dynamically.

Pei *et al.* [14] presented a dynamic service chaining model that determines whether to use VNF instances that are already running or deploy additional instances for new arrival requests. The authors presented a service chain embedding algorithm and an algorithm that releases the resources of redundant VNF instances. Li *et al.* [15] presented a model that balances the time-varying workload of service chain requests where how the workload fluctuates is already known. The authors presented a two-stage solution scheme where each request is mapped one by one and then the mapping is adjusted

TABLE I
SUMMARY OF STUDIES ON SERVICE CHAINING.

Reference	Objective	Traffic changing effects	Looping routes	Relaxing order constraints	Delay constraints	Static / dynamic	Others
Ma *et al.* [4]	Minimizing link utilization	✓		✓		Static	
Huang *et al.* [5]	Maximizing profit	✓				Static	
Huin *et al.* [6]	Minimizing link utilization		✓ (Layered network)			Static	
Allybokus *et al.* [7]	Minimizing sum of link utilization cost and VNF placement cost			✓	✓	Static	
Hyodo *et al.* [8]	Minimizing sum of link utilization cost and VNF placement cost		✓ (Layered network)	✓	✓	Static	
Chen *et al.* [9]	Minimizing sum of link utilization cost and VNF placement cost	✓		✓		Static	
Sumi *et al.* [10]	Minimizing link utilization and number of VNF instances	✓		✓		Static	
Sasabe *et al.* [12]	Minimizing total delay		✓ (Augmented network)			Static	
Xu *et al.* [13]	Minimizing setup and operation costs				✓	Static / dynamic	
Pei *et al.* [14]	Minimizing sum of resource consumption cost and VNF placement cost				✓	Dynamic	
Li *et al.* [15]	Minimizing number of activated physical machines				✓	Static	Time-varying workloads
Li *et al.* [16]	Minimizing number of used servers					Static	Availability requirements
Hawilo *et al.* [17]	Minimizing total delay				✓	Static	Availability requirements
Karimzadeh-Farshbafan *et al.* [18]	Minimizing service placement cost					Dynamic	Availability requirements
Pei *et al.* [19]	Minimizing total delay				✓	Dynamic	Deep learning
Solozabal *et al.* [20]	Minimizing power consumption				✓	Dynamic	Deep Reinforcement learning
This work	Minimizing sum of link utilization cost and VNF placement cost	✓	✓ (Layered network)	✓		Static	

so that the resource consumption is reduced. The mapping adjustment part is divided into the intra cluster adjustment and the inter cluster adjustment, where a cluster is a set of neighboring physical machines. Since these studies assume that the transmission capacity does not change along the service path, the link capacity can be overprovisioned to service chains.

The studies in [4], [5], [9], and [10] dealt with service chaining problems considering traffic chaining effects of VNFs. Ma *et al.* [4] dealt with a problem that determines the VNF placement and the VNF visit order of a single request that requires traffic-changing VNFs. The authors presented VNF placement algorithms and an algorithm that determines the VNF visit order of a request while considering the traffic changing effects of VNFs. The model in [4] does not determine the VNF placement while considering multiple requests in a time. Furthermore, request routes cannot include any loops. Huang *et al.* [5] dealt with a problem that determines the deployment of virtual machines that run VNFs on physical machines. The authors did not discuss the routing problem; the impact of the link utilization cost is ignored. Chen *et al.* [9] dealt with a problem that determines the VNF placement for multiple requests in a time where request routes are predetermined; VNFs can be deployed on only nodes that are included in the predetermined routes. Sumi *et al.* [10] dealt with a problem that determines the VNF placement, the VNF visit orders of requests, and request routes while considering multiple requests in a time. Request routes are determined by choosing the best of several candidates output by a k-shortest path algorithm [21], [22]. The model in [10] does not allow looping routes.

The studies in [16], [17], and [18] dealt with service chaining problems with availability requirements. These studies determine the resource allocation with redundancy. Li *et al.* [16] presented a service chaining model that guarantees the availability requirements while minimizing the number of activated physical machines. The authors presented an algorithm that executes the request mapping and the modification of the resource allocation so that the VNF instantiation cost is reduced without violating the availability requirements.

Hawilo *et al.* [17] presented a reliability-aware service chaining model with delay constraints. The authors presented a VNF placement algorithm based on the betweenness centrality. Karimzadeh-Farshbafan *et al.* [18] presented a dynamic reliability-aware service chaining model. The authors obtained the resource allocation with a Markov decision process. Service availabilities are not focused in our proposed model. However, the proposed model has a room of expansion by introducing availability requirements and redundant resource allocation.

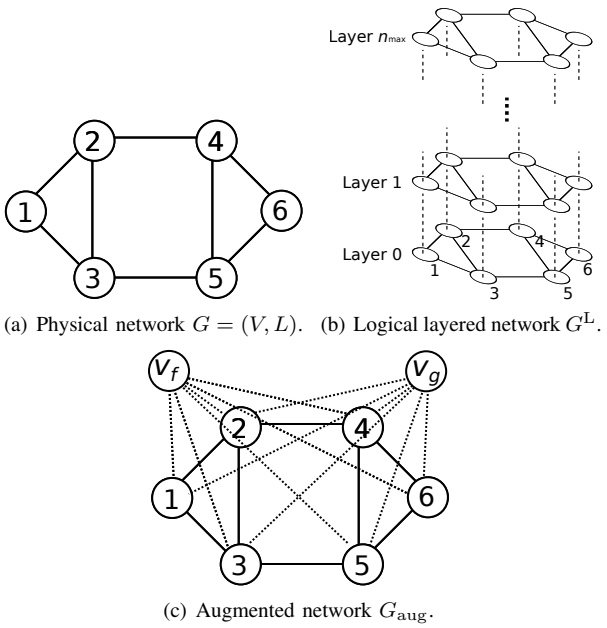
The studies in [19] and [20] presented deep learning based approaches to service chaining problems. Pei *et al.* [19] presented a deep learning based two-phase algorithm consisting of a VNF selection part and a VNF chaining part. The algorithm determines which VNF instances are used for each request and then constructs request routes according to the predefined VNF visit orders. Training data for the deep learning model are generated by running an optimal algorithm. Compared to an approach that computes an optimal solution, we can expect that a cost-efficient resource allocation is obtained in a short time with the deep learning model. Solozabal *et al.* [20] presented a deep reinforcement learning based approach for a VNF placement problem. The reinforcement learning model does not need optimal labels; the model is trained in a feedback loop where feedback signals that represent the quality of solutions are generated. Deep reinforcement learning models like that in [20] can carry on learning processes while provisioning service chains dynamically. There is a possibility that dynamic service chain provisioning based on our proposed model is realized with deep learning.

B. Relaxing routing constraints

A route of a service chain request consists of several sections: a section between the source and the first VNF node, that between each pair of nodes where consecutive VNFs are deployed, and that between the final VNF node and the destination. Packet transmission is conducted sequentially section by section, which can cause a looping route that consists of multiple sections. The network service header (NSH) [23] is attached to packets to realize service chaining. Packets are transmitted to the specified VNFs by service function forwarders, which are switches that forward arriving packets according to the NSHs of the packets.

The models in [7] and [10] determine request routes on a single-layered graph. If the physical network is represented by a single-layered graph, it is difficult to consider the case that a request route passes through the same link in the same direction more than once. There is a chance to reduce the cost of service chain provisioning by permitting such a case in determining request routes.

Huin *et al.* [6] and Hyodo *et al.* [8] introduced a logical layered network in order to allow looping routes. Figs. 2(a) and 2(b) show an example of physical network $G = (V, L)$ and that of logical layered network G^{L}, respectively. In this work, we use a logical layered network; detailed explanations of the logical layered network are described in Section III-A. Sasabe *et al.* [12] presented an augmented network. Augmented network G_{aug} is constructed by adding imaginary



(a) Physical network $G = (V, L)$. (b) Logical layered network G^{L}.

(c) Augmented network G_{aug}.

Fig. 2. Physical network, logical layered network, and augmented network.

nodes and virtual links to physical network $G = (V, L)$. Fig. 2(c) shows an example of G_{aug}, which corresponds to physical network $G = (V, L)$ shown in Fig. 2(a). The explanation of the augmented network is described in Appendix A.

Both a logical layered network and an augmented network can be applied to the service chaining problem that we deal with in this work. The number of decision variables in the ILP problem introducing a logical layered network is less than that introducing an augmented network. We discuss the number of decision variables when each network is adopted in detail in Appendix A.

C. Relaxing visit order constraints

Some VNFs have visit order constraints, but there are some cases in which changing the visit order does not impact service chain performance. For example, the request must pass through the internet protocol security (IPSec) decryptor before entering the network address translation (NAT) gateway, while the request can pass through a virtual private network (VPN) proxy either before or after a firewall [4]. We can expect more efficient service chaining if the determination of visit order is flexible.

In order to realize the relaxation of visit order constraints, Allybokus *et al.* [7] introduced the parameter representing visit order constraints among VNFs. Let F be a set of VNFs. $p_r(f, g)$ represents the visit order constraints between $f \in F$ and $g \in F$. If $f \in F$ must be processed before $g \in F$, we define $p_r(f, g)$ as $p_r(f, g) = -p_r(g, f) = 1$. If $p_r(g, f)$ equals -1, $g \in F$ must be processed after $f \in F$. If there is no constraint between $f \in F$ and $g \in F$, we define $p_r(f, g)$ as $p_r(f, g) = p_r(g, f) = 0$. The visit order parameter enables us to consider, and thus relax, visit order constraints in the optimization problem.

D. Visit order determination algorithm considering traffic changes

We can accommodate more requests in a network by minimizing, for each request, the traffic capacity reserved on each link of its route. Moreover, traffic-decreasing VNFs should be visited as early as possible, given the constraints on visit order, while traffic-increasing VNFs should be visited as late as possible on the route.

Ma *et al.* [4] presented *Converting Partially-Ordered Set to Totally-Ordered Set With Lookahead of* k, an algorithm for determining visit order considering both traffic changes and visit order constraints. This algorithm determines the visit order so as to minimize the traffic amount of a request between its source and destination while considering visit order constraints. We refer to this algorithm as Ma's algorithm hereinafter. The definition and the procedure of Ma's algorithm are described in Appendix B.

Note that Ma's algorithm is applied to only one service chain request at a time. In [4], the VNF visit order and VNF placement are determined for each request one by one. Furthermore, the work in [4] assumes that each VNF instance cannot be shared by multiple requests. This leads to an increase in the cost for the computation resource utilization. In this work, we determine VNF placement while considering multiple requests at a time and assuming that each VNF instance can be shared.

III. Proposed model

A. Definition of proposed model

We represent the physical network by bidirected graph $G = (V, L)$, where V is the set of nodes and L is the set of links. A set of requests is denoted by R. Let F be a set of VNFs and F_r be the set of VNFs that $r \in R$ requires, where $F_r \subseteq F$. The number of VNFs that $r \in R$ requires is denoted by n_r, i.e., $n_r = |F_r|$.

In order to permit a request route to pass through the same link in the same direction more than once and express of VNF traffic changes, we introduce logical layered network graph G^{L}. G^{L} is constructed by adding $n_{\max}$ replicas of $G = (V, L)$ to original graph $G = (V, L)$, where $n_{\max} = \max_{r \in R} n_r$. We call original graph $G = (V, L)$ the zeroth layer and each replica of $G = (V, L)$ the ith layer ($i = 1, 2, ..., n_{\max}$). The ith layer and the $(i+1)$th layer are connected by directed virtual links from v^i to v^{i+1}, where v^i is the node on the ith layer corresponding to $v \in V$. We put v^r_{s}, the source of $r \in R$, on the zeroth layer and v^r_{d}, the destination of $r \in R$, on the n_rth layer. The route of $r \in R$ passes through on links of the ith layer until it reaches the node where the ith required VNF of $r \in R$ is processed, and then the route uses a virtual link to move to the $(i+1)$th layer. If a request route goes through the same link on different layers, the proposed model reserves the transmission capacity for each layer. Service chains are routed in the network so that the sum of the demanded transmission capacity of each link does not exceed its maximum capacity.

We assume CPU cores as the computation resources that VNFs require. The fractional number of CPU cores that VNF $f \in F$ requires per unit of transmission capacity is denoted by Δ_f. If several requests require the same VNF, they can share the same VNF on the same CPU core as long as the total of the computation resources required by VNFs does not exceed that of the CPU core. If the sum of the fractional number of the CPU core that is used for VNF f exceeds one, additional CPU cores, which can be on a different server, are used for VNF f. Note that we can prohibit a CPU core from being shared by multiple requests by regarding the same type of VNFs for different requests as different VNFs if there is a problem in sharing the same core. For example, if requests r_1 and r_2 pass through VNF f, we can distinguish VNF f for r_1 and r_2 by designating them as f_{r_1} and f_{r_2}, respectively.

TABLE II
Descriptions of sets.

Set	Description
$G = (V, L)$	Physical network
G^{L}	Logical layered network
V	Set of nodes
L	Set of links
R	Set of requests
F	Set of VNFs
F_r	Set of VNFs that $r \in R$ requires ($F_r \subseteq F$)
$\omega^+(v)$	Set of egress flows from $v \in V$
$\omega^-(v)$	Set of ingress flows to $v \in V$
J_r	Set of visit order patterns for $r \in R$ satisfying visit order constraints

The maximum transmission capacity of link $l \in L$ is denoted by b_l. The total number of CPU cores on node $v \in V$ is denoted by c_v. h_f represents the traffic change rate of VNF f. In this model, we assume that the traffic change rate of each VNF to be constant. The initial transmission capacity of $r \in R$ is denoted by D^r_{init}. The set of egress flows from $v \in V$ and the set of ingress flows to $v \in V$ are denoted by $\omega^+(v)$ and $\omega^-(v)$, respectively.

Ψ_l represents the cost per unit of transmission capacity, which is incurred when requests pass through $l \in L$. The cost of passing through virtual links in the logical layered network, G^{L}, is set to zero. Ψ_f represents the cost per CPU core that VNF $f \in F$ utilizes. The proposed model determines request routes and VNF placement so as to minimize the sum of the cost of link utilization and the cost of VNF placement.

For easy reference, the descriptions of sets, parameters, and decision variables are summarized in Tables II, III, and IV, respectively.

B. Problem formulation

1) Basic formulation: The transmission capacity of $r \in R$ on the ith layer is denoted by decision variable δ^i_r. $\phi^{r,i}_l$ is a decision variable such that $\phi^{r,i}_l = 1$ if $r \in R$ passes through $l \in L$ on the ith layer and 0 otherwise. The number of CPU cores that VNF $f \in F$ requires on $v \in V$ is denoted by A^f_v.

The service chain provisioning problem is formulated as follows.

$$\text{minimize} \quad \sum_{r \in R} \sum_{l \in L} \sum_{i=0}^{n_r} \delta^i_r \phi^{r,i}_l \Psi_l + \sum_{v \in V} \sum_{f \in F} A^f_v \Psi_f. \qquad (1)$$

The objective function is shown in (1). The first term of (1) represents the sum of the link utilization cost. The second term

TABLE III
DESCRIPTIONS OF PARAMETERS.

Parameter	Description
n_r	Number of VNFs that $r \in R$ requires ($n_r = \|F_r\|$)
v_{s}^r	Source of $r \in R$
v_{d}^r	Destination of $r \in R$
Δ_f	Number of CPU cores that $f \in F$ requires per unit of transmission capacity
b_l	Maximum transmission capacity of $l \in L$
c_v	Total number of CPU cores on $v \in V$
h_f	Traffic change rate of $f \in F$
D_{init}^r	Initial transmission capacity of $r \in R$
Ψ_l	Link utilization cost per unit of transmission capacity for $l \in L$
Ψ_f	CPU utilization cost per core that $f \in F$ utilizes
$D_r^{i,j}$	Transmission capacity of $r \in R$ on the ith layer for $j \in J_r$
$e_f^{r,i,j}$	Binary parameter that is 1 if the ith VNF in $j \in J_r$ is $f \in F_r$; 0 otherwise

TABLE IV
DESCRIPTIONS OF DECISION VARIABLES.

Variable	Description
δ_r^i	Real number variable that represents the transmission capacity of $r \in R$ on the ith layer
$\phi_l^{r,i}$	Binary variable that is 1 if $r \in R$ passes through $l \in L$ on the ith layer; 0 otherwise
A_v^f	Integer variable that represents the number of CPU cores that $f \in F$ requires on $v \in V$
$\alpha_v^{r,i,f}$	Binary variable that is 1 if $f \in F_r$ is the ith VNF of $r \in R$ and is processed on $v \in V$; 0 otherwise
$Y_i^{r,f}$	Binary variable that is 1 if $f \in F_r$ of $r \in R$ has been passed through until the ith layer; 0 otherwise
$\pi_l^{r,i,j}$	Binary variable that is 1 if $r \in R$ passes through $l \in L$ on the ith layer with order pattern $j \in J_r$; 0 otherwise
$\alpha_v^{r,i}$	Binary variable that is 1 if the ith VNF of $r \in R$ is processed on $v \in V$; 0 otherwise
$\beta_v^{r,i,j}$	Binary variable that is 1 if the ith VNF of $r \in R$ in $j \in J_r$ is processed on $v \in V$; 0 otherwise
κ_j^r	Binary variable that is 1 if $r \in R$ uses order pattern $j \in J_r$; 0 otherwise

represents the sum of the VNF placement cost. Constraints are represented as follows.

$$\sum_{l\in\omega^+(v)} \phi_l^{r,i} - \sum_{l\in\omega^-(v)} \phi_l^{r,i} + \sum_{f\in F_r} \alpha_v^{r,i,f} - \sum_{f\in F_r} \alpha_v^{r,i-1,f} = 0, \quad \forall v \in V, r \in R, 0 < i < n_r, \tag{2}$$

$$\sum_{l\in\omega^+(v)} \phi_l^{r,0} - \sum_{l\in\omega^-(v)} \phi_l^{r,0} + \sum_{f\in F_r} \alpha_v^{r,0,f} = \begin{cases} 1 & \text{if } v = v_{\mathrm{s}}^r, \\ 0 & \text{otherwise}, \end{cases} \quad \forall v \in V, r \in R, \tag{3}$$

$$\sum_{l\in\omega^+(v)} \phi_l^{r,n_r} - \sum_{l\in\omega^-(v)} \phi_l^{r,n_r} - \sum_{f\in F_r} \alpha_v^{r,n_r-1,f} = \begin{cases} -1 & \text{if } v = v_{\mathrm{d}}^r, \\ 0 & \text{otherwise}, \end{cases} \quad \forall v \in V, r \in R. \tag{4}$$

Equations (2), (3), and (4) represent the constraints that guarantee flow conservation on the ith layer ($0 < i < n_r$), the zeroth layer, and the n_rth layer of logical layered network G^{L}, respectively. $\alpha_v^{r,i,f}$ is a decision variable such that $\alpha_v^{r,i,f} = 1$ if the ith VNF $f \in F_r$ of $r \in R$ is processed on $v \in V$ and 0 otherwise. Note that the index of required VNFs is zero-based. $\alpha_v^{r,i,f}$ also represents a flow from the ith layer to the $(i+1)$th layer. Equation (3) shows that the number of egress flows is one more than that of ingress flows on source v_{s}^r. Equation (4) shows that the number of ingress flows is one more than that of egress flows on destination v_{d}^r.

$$\sum_{r\in R} \sum_{i=0}^{n_r} \delta_r^i \phi_l^{r,i} \le b_l, \forall l \in L, \tag{5}$$

$$\sum_{f\in F} A_v^f \le c_v, \forall v \in V. \tag{6}$$

Equation (5) guarantees that the sum of the transmission capacity demanded of $l \in L$ does not exceed maximum transmission capacity b_l. Equation (6) guarantees that the sum of CPU cores that VNFs require on $v \in V$ does not exceed total number of CPU cores c_v.

$$Y_0^{r,f} = 0, \forall r \in R, f \in F, \tag{7}$$

$$Y_{n_r}^{r,f} = 1, \forall r \in R, f \in F_r, \tag{8}$$

$$Y_i^{r,f} - Y_{i-1}^{r,f} = \sum_{v\in V} \alpha_v^{r,i-1,f}, \forall r \in R, f \in F_r, 0 < i < n_r, \tag{9}$$

$$(Y_i^{r,f} - Y_i^{r,g}) p_r(f,g) \ge 0, \forall r \in R, f, g \in F_r, 0 < i < n_r. \tag{10}$$

Equations (7)–(10) represent visit order constraints. $Y_i^{r,f}$ is a decision variable such that $Y_i^{r,f} = 1$ if $f \in F_r$ of $r \in R$ has been passed through until the ith layer and 0 otherwise. Equation (7) means that no VNF is processed on the zeroth layer before reaching the node on which the first VNF is processed. Equation (8) guarantees that all the VNFs required have been processed before reaching the n_rth layer. In (7), $Y_i^{r,f} - Y_{i-1}^{r,f} = 1$ if $(i-1)$th VNF of $r \in R$ is $f \in F_r$ and 0 otherwise. In (10), $Y_i^{r,f}$ is forced to be equal to or more than $Y_i^{r,g}$ if $f \in F$ needs to be processed before $g \in F$, i.e., $p_r(f,g) = 1$. Equation (10) holds in all cases if there are no constraints between $f \in F$ and $g \in F$, i.e., $p_r(f,g) = 0$.

$$\delta_r^0 = D_{\mathrm{init}}^r, \forall r \in R, \tag{11}$$

$$\delta_r^i = \delta_r^{i-1} \sum_{f\in F_r} h_f (Y_i^{r,f} - Y_{i-1}^{r,f}), \forall r \in R, 0 < i \le n_r. \tag{12}$$

Equations (11) and (12) determine the traffic amount of $r \in R$ on the zeroth layer and the ith layer ($0 < i \le n_r$), respectively. Equation (11) means that the traffic amount of $r \in R$ on the zeroth layer is equal to initial transmission capacity D_{init}^r. Equation (12) means that the traffic amount on the ith layer is the product of the traffic amount on the $(i-1)$th layer and the traffic change rate of the $(i-1)$th VNF.

$$\sum_{i=0}^{n_r} \sum_{v\in V} \alpha_v^{r,i,f} = 1, \forall f \in F_r, r \in R. \tag{13}$$

Equation (13) guarantees that $f \in F_r$ is processed on only one node in the network for $r \in R$.

$$\sum_{r\in R} \sum_{i=0}^{n_r-1} \Delta_f \delta_r^i \alpha_v^{r,i,f} \le A_v^f, \forall f \in F, v \in V. \tag{14}$$

Equation (14) determines the number of CPU cores that $f \in F$ utilizes on $v \in V$.

2) Formulation in ILP form: As the objective function and some constraints involve the product of δ_r^i and binary variables, the problem formulated in (1)–(14) is non-linear. In order to formulate the problem as an ILP problem, we write the transmission capacity as constant $D_r^{i,j}$. $D_r^{i,j}$ can be obtained by calculating the transmission capacity on the ith layer for each visit order pattern $j \in J_r$, where J_r is the set of visit order patterns for $r \in R$ satisfying visit order constraints; $D_r^{i,j} = h_f D_r^{i-1,j}$ if $f \in F$ is the $(i-1)$th VNF of $r \in R$ for $j \in J_r$. Note that we can also handle the case where the traffic change rate is a non-linear function of the transmission capacity; $D_r^{i,j} = h_{\mathrm{nlf}}^f(D_r^{i-1,j})$, where $h_{\mathrm{nlf}}^f(\cdot)$ is a non-linear function that represents the traffic change rate of $f \in F$, if $f \in F$ is the $(i-1)$th VNF of $r \in R$ for $j \in J_r$. $e_f^{r,i,j}$ is a parameter such that $e_f^{r,i,j} = 1$ if the ith VNF in order pattern $j \in J_r$ is $f \in F_r$ and 0 otherwise. $\pi_l^{r,i,j}$ is a decision variable such that $\pi_l^{r,i,j} = 1$ if $r \in R$ passes through $l \in L$ on the ith layer with order pattern $j \in J_r$ and 0 otherwise. $\alpha_v^{r,i}$ is a decision variable such that $\alpha_v^{r,i} = 1$ if the ith VNF of $r \in R$ is processed on $v \in V$ and 0 otherwise. $\beta_v^{r,i,j}$ is a decision variable such that $\beta_v^{r,i,j} = 1$ if the ith VNF of $r \in R$ in order pattern $j \in J_r$ is processed on $v \in V$ and 0 otherwise. κ_j^r is a decision variable such that $\kappa_j^r = 1$ if r uses order pattern $j \in J_r$ and 0 otherwise.

The proposed model can be formulated as an ILP problem as follows.

$$\text{minimize} \quad \sum_{r\in R}\sum_{l\in L}\sum_{i=0}^{n_r}\sum_{j\in J_r} D_r^{i,j}\pi_l^{r,i,j}\Psi_l + \sum_{v\in V}\sum_{f\in F} A_v^f \Psi_f. \tag{15}$$

The objective function is shown in (15), which corresponds to (1). Constraints are represented as follows.

$$(6), \tag{16}$$

$$\sum_{l\in\omega^+(v)} \phi_l^{r,i} - \sum_{l\in\omega^-(v)} \phi_l^{r,i} + \alpha_v^{r,i} - \alpha_v^{r,i-1} = 0, \quad \forall v \in V, r \in R, 0 < i < n_r, \tag{17}$$

$$\sum_{l\in\omega^+(v)} \phi_l^{r,0} - \sum_{l\in\omega^-(v)} \phi_l^{r,0} + \alpha_v^{r,0} = \begin{cases} 1 & \text{if } v = v_{\mathrm{s}}^r, \\ 0 & \text{otherwise}, \end{cases} \quad \forall v \in V, r \in R, \tag{18}$$

$$\sum_{l\in\omega^+(v)} \phi_l^{r,n_r} - \sum_{l\in\omega^-(v)} \phi_l^{r,n_r} - \alpha_v^{r,n_r-1} = \begin{cases} -1 & \text{if } v = v_{\mathrm{d}}^r, \\ 0 & \text{otherwise}, \end{cases} \quad \forall v \in V, r \in R, \tag{19}$$

$$\sum_{r\in R}\sum_{i=0}^{n_r}\sum_{j\in J_r} D_r^{i,j}\pi_l^{r,i,j} \le b_l, \forall l \in L, \tag{20}$$

$$\sum_{r\in R}\sum_{i=0}^{n_r-1}\sum_{j\in J_r} \Delta_f D_r^{i,j} e_f^{r,i,j}\beta_v^{r,i,j} \le A_v^f, \forall f \in F, v \in V. \tag{21}$$

Equations (17)–(19) correspond to (2)–(4). Equation (20) corresponds to (5). Equation (21) corresponds to (14).

$$\sum_{j\in J_r} \kappa_j^r = 1, \forall r \in R. \tag{22}$$

Equation (22) guarantees that $r \in R$ uses only one order pattern j in J_r.

$$1 - \phi_l^{r,i} - \kappa_j^r + \pi_l^{r,i,j} \ge 0, \forall l \in L, r \in R, j \in J_r, 0 \le i \le n_r, \tag{23}$$

$$\phi_l^{r,i} - \pi_l^{r,i,j} \ge 0, \forall l \in L, r \in R, j \in J_r, 0 \le i \le n_r, \tag{24}$$

$$\kappa_j^r - \pi_l^{r,i,j} \ge 0, \forall l \in L, r \in R, j \in J_r, 0 \le i \le n_r, \tag{25}$$

$$1 - \alpha_v^{r,i} - \kappa_j^r + \beta_v^{r,i,j} \ge 0, \forall v \in V, r \in R, j \in J_r, 0 \le i \le n_r, \tag{26}$$

$$\alpha_v^{r,i} - \beta_v^{r,i,j} \ge 0, \forall v \in V, r \in R, j \in J_r, 0 \le i \le n_r, \tag{27}$$

$$\kappa_j^r - \beta_v^{r,i,j} \ge 0, \forall v \in V, r \in R, j \in J_r, 0 \le i \le n_r. \tag{28}$$

Equations (23)–(25) and (26)–(28) yield $\pi_l^{r,i,j} = \phi_l^{r,i}\kappa_j^r$ and $\beta_v^{r,i,j} = \alpha_v^{r,i}\kappa_j^r$ in linear equations, respectively. The number of constraints in (15)–(28) is $|V|(\sum_{r\in R}(3|J_r|+1)(n_r+1)+|F|+1)+|L|(3\sum_{r\in R}|J_r|(n_r+1)+1)+|R|$. The number of decision variables in (15)–(28) is $|V|(\sum_{r\in R}(|J_r|+1)(n_r+1)+|F|)+|L|(\sum_{r\in R}(|J_r|+1)(n_r+1))+\sum_{r\in R}|J_r|$.

3) Special case where $|J_r| = 1$: If the visit order of each request is predetermined, i.e., $|J_r| = 1$, decision variables such as κ_j^r and constraints such as (22)–(28) for determining the visit order are not needed. Therefore, the ILP model in (15)–(28) can be simplified in this case. D_r^i represents the transmission capacity of $r \in R$ on the ith layer. $e_f^{r,i}$ is a parameter such that $e_f^{r,i} = 1$ if the ith VNF of $r \in R$ is $f \in F_r$ and 0 otherwise. An ILP model for requests whose visit orders are predetermined is formulated as follows:

$$\text{minimize} \quad \sum_{r\in R}\sum_{l\in L}\sum_{i=0}^{n_r} D_r^i \phi_l^{r,i}\Psi_l + \sum_{v\in V}\sum_{f\in F} A_v^f \Psi_f, \tag{29}$$

$$(6), (17)\text{–}(19), \tag{30}$$

$$\sum_{r\in R}\sum_{i=0}^{n_r} D_r^i \phi_l^{r,i} \le b_l, \forall l \in L, \tag{31}$$

$$\sum_{r\in R}\sum_{i=0}^{n_r-1} \Delta_f D_r^i e_f^{r,i}\alpha_v^{r,i} \le A_v^f, \forall f \in F, v \in V. \tag{32}$$

The number of constraints in (29)–(32) is $|V|(\sum_{r\in R}(n_r+1)+|F|+1)+|L|$. The number of decision variables in (29)–(32) is $|V|(\sum_{r\in R} n_r + |F|) + |L|(\sum_{r\in R}(n_r+1))$.

C. Problem hardness analysis

We analyze the hardness of the service chaining problem considering traffic changes due to VNFs (SCP-TCV), which is formulated in the ILP problem in (15)–(28). We define the decision version of SCP-TCV (SCP-TCV-D) as follows.

Definition 1. Bidirected graph $G = (V, L)$, set of requests R, and set of VNFs F are given. Logical layered network G^{L} is constructed from $G = (V, L)$. The number of layers in G^{L} is $n_{\max} = \max_{r\in R} n_r$, where n_r is the number of required VNFs for $r \in R$. The maximum transmission capacity of $l \in L$ is b_l. The total number of CPU cores on $v \in V$ is c_v. The source and destination nodes of $r \in R$ are $v_{\mathrm{s}^r} \in V$ and $v_{\mathrm{d}^r} \in V$, respectively. $r \in R$ must pass through VNFs in $F_r \subseteq F$. A

set of possible VNF visit order patterns for $r \in R$ is given as J_r. The transmission capacity of $r \in R$ on the ith layer of G^{L} for $j \in J_r$ is $D_r^{i,j}$. $f \in F$ requires Δ_f CPU cores per unit of transmission capacity. The link utilization cost per unit of transmission capacity for $l \in L$ is Ψ_l. The CPU utilization cost per core for $f \in F$ is Ψ_f. Is there any resource allocation such that the total cost is T or less?

Theorem 1. SCP-TCV-D is NP-complete.

Proof. First, we show that SCP-TCV-D is in NP. Let $J_{\max}$ be the maximum number of VNF order visit patterns, i.e., $J_{\max} = \max_{r \in R} |J_r|$. The time complexities of confirming whether each constraint is satisfied are as follows:

- Flow conservation: $O(n_{\max}|R||V||L|)$
- Link capacity: $O(J_{\max} n_{\max}|R||L|)$
- Node capacity: $O(|V||F|)$
- Determining required number of CPU cores for each VNF on each node: $O(J_{\max} n_{\max}|R||V||F|)$
- Selecting VNF visit order pattern: $O(J_{\max}|R|)$
- Associating VNF visit order pattern with link utilization: $O(J_{\max} n_{\max}|R||L|)$
- Associating VNF visit order pattern with VNF placement: $O(J_{\max} n_{\max}|R||V|)$

The time complexity of checking whether the total cost is T or less is $O(J_{\max} n_{\max}|R||L| + |V||F|)$. Therefore, SCP-TCV-D is in NP.

Second, we prove that the decision version of the bin packing problem (BPP-D), which is NP-complete [24], [25], is reducible to SCP-TCV-D. Let H a set of items. BPP-D decides whether there is an allocation of items to a finite number of bins such that the sum of the sizes of items in each bin does not exceed the capacity of the bin, where the size of item $h \in H$, the number of bins, and the capacity of each bin are w_h, K, and B, respectively.

We construct an SCP-TCV-D instance from any BPP-D instance in the following steps:

1) Connected bidirected graph $G = (V, L)$ is given, where the number of node is K, i.e., $|V| = K$. Logical layered network G^L is constructed from $G = (V, L)$.
2) The number of requests is the same with the number of items in the corresponding BPP-D instance, i.e., $|R| = |H|$. There is a one-to-one correspondence between requests in SCP-TCV-D and items in BPP-D. The item that corresponds to $r \in R$ is denoted as $r \in H$ hereafter.
3) The link utilization cost is set to zero, i.e., $\Psi_l = 0, \forall l \in L$.
4) Each link can accommodate all requests no matter how many times each request passes through the link, i.e., $b_l = \sum_{r \in R} \sum_{i=0}^{n_r} \sum_{j \in J_r} D_r^{i,j}, \forall l \in L$.
5) All requests pass through only one particular VNF, namely f, i.e., $F_r = \{f\} = F, \forall r \in R$. From this, the number of VNF visit order pattern is one, i.e., $|J_r| = 1, \forall r \in R$. In addition, G^L consists of two layers, i.e., $i \in \{0, 1\}$. VNF f does not change the transmission capacity.
6) The transmission capacity of each request is set to the size of the corresponding item, i.e., $D_r^{i,j} = w_r, \forall r \in R, i \in \{0, 1\}$.
7) The source and destination nodes of each request are arbitrarily selected from nodes in V.
8) The CPU utilization cost of VNF f is set to one, i.e., $\Psi_f = 1$.
9) The number of CPU cores that VNF f requires per unit of transmission capacity is set to $1/B$, i.e., $\Delta_f = 1/B$; the sum of the transmission capacity that is processed by VNF f cannot exceed B.
10) Each node has only one CPU core, i.e., $c_v = 1, \forall v \in V$. From this, $|V| = K$, and $\Delta_f = 1/B$, there is a one-to-one correspondence between nodes in SCP-TCV-D and bins in BPP-D.
11) The threshold of the total cost is set to K, i.e., $T = K$.

The time complexity of the construction of an SCP-TCV-D instance is $O(J_{\max} n_{\max}|R||L| + |V|)$.

If a BPP-D instance is a Yes instance, a given set of items can be accommodated in K bins or less; there is an item allocation such that $\sum_{h \in H_p} w_h \leq B, \forall p \in P, \bigsqcup_{p \in P} H_p = H$, where P and H_p are a set of K bins and a set of items packed in $p \in P$, respectively. Note that $H_p = \emptyset, \exists p \in P$ if a set of items is accommodated in less than K bins. From the one-to-one correspondence between requests and items and that between nodes and bins, there is a VNF placement such that $\sum_{r \in R_v} \sum_{i=0}^{n_r - 1} \sum_{j \in J_r} D_r^{i,j} = \sum_{r \in R_v} w_r \leq B, \forall v \in V, \bigsqcup_{v \in V} R_v = R$, where R_v is a set of requests that passes through VNF f on $v \in V$ in the SCP-TCV-D instance; this ensures that there is a VNF placement such that the number of used CPU cores is K or less since the transmission capacity of $r \in R$, the number of nodes, and the maximum accommodatable transmission capacity of $v \in V$ are w_r, K and B, respectively. The route of $r \in R$ can be constructed by finding two paths, one is that between the source and v_r and the other is that between v_r and the destination, where v_r is the node where VNF f is running for r. From this, there is a resource allocation such that the number of used CPU cores can be K or less. The total cost is K or less since $\Psi_l = 0, \forall l \in L$ and $\Psi_f = 1$. As the threshold of the total cost is set to K, the SCP-TCV-D instance is a Yes instance.

Conversely, if the constructed SCP-TCV-D instance is a Yes instance, the total cost is K or less. As the node capacity constraints are satisfied, $\sum_{r \in R_v} \sum_{i=0}^{n_r - 1} \sum_{j \in J_r} D_r^{i,j} = \sum_{r \in R_v} w_r \leq B, \forall v \in V, \bigsqcup_{v \in V} R_v = R$. Note that $R_v = \emptyset, \exists v \in V$ if the number of used CPU cores is less than K. From the one-to-one correspondence between requests and items and that between nodes and bins, there is an item allocation such that $\sum_{h \in H_p} w_h \leq B, \forall p \in P, \bigsqcup_{p \in P} H_p = H$ in the corresponding BPP-D instance; this ensures that there is no item that cannot be packed in any bin in P, where the size of item $h \in H$, the number of bins, and the capacity of each bin are w_h, K, and B, respectively. Therefore, there is an item allocation to K bins or less; the corresponding BPP-D instance is a Yes instance.

From the discussion above, BPP-D is reducible to SCP-TCV-D; any problem in NP is reducible to SCP-TCV-D since

BPP-D is NP-complete. As SCP-TCV-D is in NP, SCP-TCV-D is NP-complete. □

IV. Methods of selecting visit order patterns

A. Overview

If in solving the ILP problem of (15)–(28) we consider all possible visit order patterns, it is difficult to obtain the routes and VNF placement in a practical time. We introduce some methods of selecting visit order patterns in order to shorten the computation time while keeping the total costs as close to the optimal value as possible.

The sum of the traffic amount reserved by each service chain request on its route can be minimized by configuring the route of each request so that it preferentially passes through VNFs with low traffic change rates while satisfying visit order constraints; this leads to a reduction in the link utilization cost. Moreover, computation resources that VNFs require can be reduced by minimizing the sum of the traffic amount reserved by each request on its route; this leads to a reduction in the VNF placement cost. In order to determine the VNF visit order of each request so as to suppress the traffic amount in the whole network, we apply Ma's algorithm [4], which is described in Appendix B.

This section introduces three methods of selecting visit order patterns based on Ma's algorithm. First, Section IV-B explains a method of solving the ILP problem in (15)–(28) with a limited number of order patterns. This method selects the visit order patterns so as to reduce the total cost while considering both routes and VNF placement. Section IV-C explains the second and the third methods, which predetermine the visit order of each request. Section IV-C1 explains a method of predetermining the visit order of each request with Ma's algorithm by setting the same value of k for all requests. Section IV-C2 explains a method of predetermining the visit order of each request with an ILP model that selects the visit order so as to suppress the number of CPU cores used.

B. Solving ILP model with limited number of order patterns

This method first obtains k_{M} order patterns for each request by setting the value of parameter k from 1 to k_{M} in Ma's algorithm. We can set k_{M} to an integer from 1 to $k_{\max}$, where $k_{\max}$ is the maximum size of the trees representing dependency relations among VNFs. Note that $k_{\max}$ is less than or equal to $n_{\max}$, that is, the maximum number of required VNFs. Let J_r be a set of k_{M} order patterns obtained with the algorithm for $r \in R$. We can solve the ILP problem in (15)–(28) by considering only k_{M} order patterns for each request. Thus, we can expect shorter computation times than needed when considering all possible order patterns. The total cost can decrease as k_{M} increases since there can be more efficient combinations of visit order patterns than the case where k_{M} is set to a smaller number. On the other hand, the computation time can increase since the number of decision variables also increases.

C. Solving ILP model with predetermined visit order

This section introduces the second and the third methods of selecting visit order patterns. These methods heuristically predetermine the visit order of each request with Ma's algorithm. Since the number of visit order patterns is only one for each request, we obtain request routes and VNF placement by solving the ILP model in (29)–(32), which is a simplified form of the ILP problem in (15)–(28).

1) Visit order predetermination with same value of k for all requests: This method uses Ma's algorithm and applies the same value of k to all requests. The total cost is obtained by solving the ILP model in (29)–(32) with the predetermined order obtained with the algorithm for each request. We can expect the traffic amount of each request to be suppressed by increasing k.

2) Visit order predetermination with ILP model: This method selects the visit order from several candidates for each request. It may be possible to obtain lower cost by using several values of k in Ma's algorithm than that achieved by applying the same value of k for all requests. We determine order patterns so as to reduce the number of VNF instances, which can lead to the reduction of the total cost.

This method first obtains $k_{\max}$ order patterns for each request by setting the value of parameter k from 1 to $k_{\max}$ in Ma's algorithm. Let J_r be a set of $k_{\max}$ order patterns obtained with the algorithm for $r \in R$. Then, the visit order for each request is selected by solving the optimization problem that minimizes the number of used CPU cores, which is defined as follows.

$$\text{minimize} \quad \sum_{f \in F} A_f. \tag{33}$$

The objective function is shown in (33). A_f represents the number of CPU cores that $f \in F$ requires. Constraints are represented as follows.

$$(22), \tag{34}$$

$$\sum_{r \in R} \sum_{i=0}^{n_r - 1} \sum_{j \in J_r} \Delta_f D_r^{i,j} e_f^{r,i,j} \kappa_j^r \leq A_f, \forall f \in F. \tag{35}$$

Equation (35) determines the number of CPU cores that $f \in F$ requires, i.e., A_f.

In order to obtain the visit order patterns that minimize the minimum necessary number of used CPU cores, we assume that all VNFs run on one particular node in this method. Note that the actual VNF placement is determined by solving the ILP problem in (29)–(32) after determining the visit orders with this method; VNF instances can be distributed on different nodes in the actual VNF placement. By applying the visit order patterns obtained with this method, we can expect that the CPU utilization cost can be suppressed. In addition, by suppressing the number of used CPU cores, we can expect that the link utilization cost can be suppressed since the number of used CPU cores depends on the transmission capacity of each request; if the number of used CPU cores decreases, the transmission capacity of each request is expected to decrease.

V. Heuristic algorithms

As the problem size increases, the ILP problems in (15)–(28) and (29)–(32) also become difficult to be solved in a practical time regardless of limiting the number of VNF visit order patterns. We introduce two heuristic algorithms for the proposed model. Section V-A explains a greedy algorithm that iteratively solves the optimization problem for a limited number of requests. Section V-B explains a heuristic algorithm that divides the problem into the VNF placement part and the routing part. We call this algorithm a problem dividing (PD) algorithm hereinafter.

A. Greedy algorithm

The greedy algorithm is illustrated in Algorithm 1. We call one execution of the while loop in lines 2–12 of Algorithm 1 an iteration here. First, the algorithm determines the VNF visit order for each request with Ma's algorithm and the ILP model in (33)–(35) as illustrated in Section IV-C2. The algorithm solves the ILP model in (29)–(32) for n_{grd} requests ($1 \leq n_{\mathrm{grd}} \leq |R|$) picked from R in each iteration. If the number of the remaining requests is less than n_{grd}, the algorithm solves the ILP model in (29)–(32) for all remaining requests. Link capacities, link utilization cost, and CPU utilization for each node and VNF are updated after solving the ILP model in (29)–(32) in each iteration. After all iterations are completed, the algorithm outputs request routes, VNF placement, and the total cost.

The greedy algorithm solves the ILP problem in (29)–(32), whose decision version is NP-complete, $\lceil |R|/n_{\mathrm{grd}} \rceil$ times. The number of constraints and that of decision variables of the ILP model in (29)–(32) in each iteration are less than or equal to those in the case where all requests are given to the ILP model in (29)–(32) at a time since $n_{\mathrm{grd}} \leq |R|$. The number of possible value patterns of integer variables in (29)–(32) exponentially increases according to the number of requests. Therefore, we can expect that the computation time is shortened by iteratively solving the ILP problem in (29)–(32) with a limited number of requests.

Algorithm 1 Greedy algorithm

Input: Network topology G, set of requests R, set of VNFs F, and n_{grd}
Output: Request routes, VNF placement, and total cost

1: Determine the VNF visit order for each request $r \in R$ with Ma's Algorithm and the ILP model in (33)–(35)
2: **while** $R \neq \emptyset$ **do**
3: **if** $n_{\mathrm{grd}} \leq |R|$ **then**
4: Let R' to be a set of n_{grd} requests picked from R
5: **else**
6: $R' \leftarrow R$
7: **end if**
8: Determine routes for $r \in R'$ and VNF placement with the ILP model in (29)–(32)
9: Update link capacities and link utilization cost
10: Update CPU utilization for each node and VNF
11: $R \leftarrow R \setminus R'$
12: **end while**

B. PD algorithm

The PD algorithm determines the resource allocation by solving two problems: the VNF placement problem and the routing problem. Note that the routing problem is solved according to the VNF placement determined with the VNF placement problem.

The VNF placement problem in the PD algorithm (VPP-PD) is formulated in an ILP problem as follows.

$$\text{minimize} \quad \sum_{r\in R}\sum_{v\in V}\sum_{i=0}^{n_r-1}(d(v_{\mathrm{s}}^r, v) + d(v_{\mathrm{d}}^r, v))\alpha_v^{r,i}. \tag{36}$$

$d(u, v)$ is the minimum number of hops between nodes $u \in V$ and $v \in V$. $d(v_{\mathrm{s}}^r, v)\alpha_v^{r,i}$ represents the number of the hops between the source and the node where the ith VNF of $r \in R$ is placed. $d(v_{\mathrm{d}}^r, v)\alpha_v^{r,i}$ represents the number of the hops between the destination and the node where the ith VNF of $r \in R$ is placed. Constraints are represented as follows.

$$\text{(6), (32),} \tag{37}$$

$$\sum_{v\in V}\alpha_v^{r,i} = 1, \forall r \in R, 0 \leq i < n_r, \tag{38}$$

$$\sum_{f\in F}\sum_{v\in V}A_v^f \leq I. \tag{39}$$

Equation (38) guarantees that the ith VNF of $r \in R$ is placed on any one node in the network. Equation (39) limits the number of CPU cores used by VNFs. We obtain the cost of VNF placement, T_{VNF}, from a solution of (36)–(39). The number of constraints in (36)–(39) is $|V|(|F|+1) + \sum_{r\in R} n_r + 1$. The number of decision variables in (36)–(39) is $|V|(\sum_{r\in R} n_r + |F|)$. The decision version of VPP-PD is NP-complete, which is shown in Appendix C.

The routing problem in the PD algorithm (RP-PD) is formulated in an ILP problem as follows.

$$\text{minimize} \quad \sum_{r\in R}\sum_{l\in L}\sum_{i=0}^{n_r} D_r^i \phi_l^{r,i}\Psi_l. \tag{40}$$

The objective function is shown in (40). This corresponds to the first term of (29); it represents the sum of the link utilization cost. Constraints are represented as follows.

$$\text{(17)–(19), (31),} \tag{41}$$

$$\alpha_{v_{f_i^r}}^{r,i} = 1, \forall r \in R, 0 \leq i < n_r. \tag{42}$$

$v_{f_i^r}$ is the node where the ith VNF of $r \in R$ is placed, which is determined in the VNF placement problem in (36)–(39). Equation (42) guarantees that $r \in R$ passes through its ith VNF on $v_{f_i^r} \in V$. The number of constraints in (40)–(42) is $|V|\sum_{r\in R}(n_r+1) + |L|\sum_{r\in R} n_r$. The number of decision variables in (40)–(42) is $|V|\sum_{r\in R} n_r + |L|\sum_{r\in R}(n_r+1)$. The decision version of RP-PD is NP-complete, which is shown in Appendix C.

The procedure of the PD algorithm is illustrated in Algorithm 2. First, the algorithm determines the VNF visit order

Algorithm 2 Problem dividing (PD) algorithm

Input: Network topology G, set of requests R, and set of VNFs F

Output: Request routes, VNF placement, and total cost T

1: Determine the VNF visit order for each request $r \in R$ with Ma's algorithm and the ILP model in (33)–(35)
2: $I \leftarrow$ optimal value obtained with the ILP model in (33)–(35)
3: $T \leftarrow \infty$
4: **while** $I \leq \sum_{v \in V} c_v$ **do**
5: Determine VNF placement by solving VPP-PD
6: $T_{\mathrm{VNF}} \leftarrow$ Cost of VNF placement obtained by solving VPP-PD
7: Determine request routes by solving RP-PD
8: $T_{\mathrm{link}} \leftarrow$ Cost of link utilization obtained by solving RP-PD
9: $T_{\mathrm{cur}} \leftarrow T_{\mathrm{VNF}} + T_{\mathrm{link}}$
10: **if** $T < T_{\mathrm{cur}}$ **then**
11: Break
12: **end if**
13: $I \leftarrow I + 1$
14: **end while**

for each request with Ma's algorithm and the ILP model in (33)–(35) as illustrated in Section IV-C2. Let I be the maximum number of CPU cores that VNFs use. We set I to be the optimal value obtained with the ILP model in (33)–(35). Then, the algorithm determines the VNF placement and request routes. We call one execution of the while loop in lines 4–14 of Algorithm 2 an iteration here.

In each iteration, the PD algorithm first determines VNF placement by solving VPP-PD, i.e., the ILP problem in (36)–(39). After the VNF placement is obtained, the PD algorithm determines request routes by solving RP-PD, i.e., the ILP problem in (40)–(42). After request routes are determined, we obtain T_{cur}, the total cost in the current iteration. If T, the total cost in the previous iteration, is less than T_{cur}, the algorithm outputs T as the total cost. Similarly, the algorithm outputs the VNF placement and request routes obtained in the previous iteration. If T_{cur} is less than or equal to T, Algorithm 2 substitutes T_{cur} for T. Then, I is incremented and the algorithm goes to the next iteration.

In addition, we introduce a modified variation of the PD algorithm. We call this variation PD considering traffic changes (PD-TC). In PD-TC, the objective function of the VNF placement problem in (36) is replaced with the following one.

$$\text{minimize} \quad \sum_{r \in R} \sum_{v \in V} \sum_{i=0}^{n_r - 1} (d(v_{\mathrm{s}}^r, v) q_{\mathrm{s}}^{r,i} + d(v_{\mathrm{d}}^r, v) q_{\mathrm{d}}^{r,i}) \alpha_v^{r,i}. \tag{43}$$

$q_{\mathrm{s}}^{r,i}$ and $q_{\mathrm{d}}^{r,i}$ are binary parameters. Let h_r^i be the traffic change rate of the ith VNF of $r \in R$. If $\iota = \mathrm{argmin}_{0 \leq \iota \leq n_r - 1} \prod_{i=0}^{\iota} h_r^i$ for $r \in R$, $q_{\mathrm{s}}^{r,i} = 1$ and $q_{\mathrm{d}}^{r,i} = 0$ for $0 \leq i \leq \iota$, and $q_{\mathrm{s}}^{r,i} = 0$ and $q_{\mathrm{d}}^{r,i} = 1$ for $\iota < i < n_r$. $d(v_{\mathrm{s}}^r, v) q_{\mathrm{s}}^{r,i} \alpha_v^{r,i}$ represents the number of the hops between the source and the node where the ith VNF of $r \in R$ is placed ($0 \leq i \leq \iota$). $d(v_{\mathrm{d}}^r, v) q_{\mathrm{d}}^{r,i} \alpha_v^{r,i}$ represents the number of the hops between the destination and the node where the ith VNF of $r \in R$ is placed ($\iota < i < n_r$). We can expect that the traffic amount decreases on nodes that are near the source and increases on nodes that are near the destination.

The PD algorithm includes solving VPP-PD and RP-PD, whose decision versions are NP-complete, at most $\sum_{v \in V} c_v$ times. However, the sizes of VPP-PD and RP-PD are small compared to SCP-TCV where $|J_r| = 1$. The number of constraints and that of decision variables of VPP-PD is smaller by $(|V|-1)\sum_{r \in R} n_r + |L| + |V||R| - 1$ and $|L| \sum_{r \in R} (n_r + 1)$, respectively, than SCP-TCV where $|J_r| = 1$. The number of constraints and that of decision variables of RP-PD are smaller by $|V|(|F|+1) + |L|(1 - \sum_{r \in R} n_r)$ and $|V||F|$, respectively, than SCP-TCV where $|J_r| = 1$.

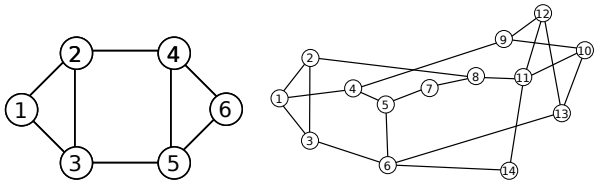
VI. Numerical results

In this section, we evaluate the performance of the proposed model in terms of the computation time and the total cost. Section VI-A shows the comparison among the proposed model and benchmarks based on existing works. Section VI-B shows the comparison between the case considering all the possible VNF visit order patterns and those limiting the number of order patterns considered in the ILP model. Section VI-C shows the evaluation of heuristic algorithms. In Sections VI-A–VI-C, we evaluate the basic performances of the proposed model and heuristic algorithms by using artificial values of parameters. Section VI-D shows the evaluation in realistic situations where the parameters of requests and VNFs are set based on data shown in [6], [12], [26]–[32].

A. Comparison with existing works

This section compares the performance of the proposed model and those of benchmarks based on existing works. One benchmark is a model that assumes the traffic amount to be constant between the endpoints, which is based on the models in [6]–[8]. We call this benchmark a constant traffic (CT) model hereafter. In the CT model, the transmission capacity of each request needs to be set to the maximum value in the traffic changes due to the required VNFs. This means that the CT model reserves the transmission capacity of each request more than actually needed when any VNF changes the traffic amount. The other is a model that considers traffic changes and selects request routes from among candidates obtained with a k-shortest path algorithm, which is based on the model in [10]. The k-shortest path algorithm is applied individually to each request since the source and destination can vary among requests. We call this benchmark a k-shortest path algorithm based (KSP) model hereafter.

We use two network topologies in this evaluation. One is a network topology that has six nodes and eight links as shown in Fig. 3(a). This network topology is called the 6-node 8-link network hereafter. The other is NSFNET [33], which is shown in Fig. 3(b). We assume a set of VNFs $F = \{f_0, f_1, f_2, f_3, f_4\}$. The traffic change rates of f_0, f_1, f_2, f_3, and f_4, are 0.5, 0.7, 1.0, 1.5, and 2.0, respectively. Each request randomly chooses at least three and at most five



(a) 6-node 8-link network. (b) NSFNET.

Fig. 3. Network topologies.

VNFs in F without overlap. Visit order constraints are set as follows:

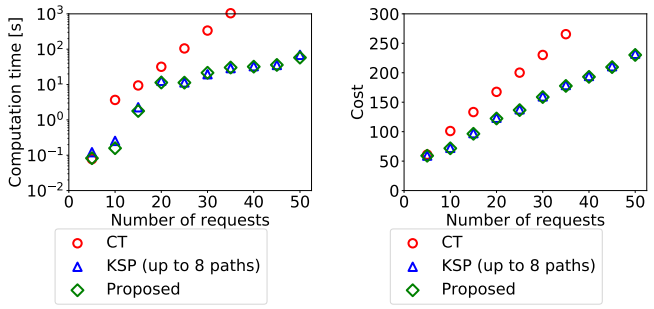
- f_0 has to be passed through after f_2 if both of them are required.
- f_4 has to be passed through after f_1 if both of them are required.

Visit order of each request is determined with Ma's algorithm with $k = 1$.

The maximum transmission capacity of each link is 200, i.e., $b_l = 200$. The link utilization cost of each link is 1 per unit of transmission capacity, i.e., $\Psi_l = 1$. The number of CPU cores attached to each node is 20, i.e., $c_v = 20$. The VNF placement cost of each VNF is 10, i.e., $\Psi_f = 10$. The number of CPU cores that each VNF requires per unit of transmission capacity is 0.1, i.e., $\Delta_f = 0.1$. The initial transmission capacity of each request is 1. The total cost in the case of considering traffic amount changes is obtained by solving the ILP model in (29)–(32) as it is. For the case of constant traffic amount, the total cost is obtained by solving an ILP model that corresponds to (29)–(32); the transmission capacity of $r \in R$ is represented by D_r, where $D_r = \max_{0 \le i \le n_r} D_r^i$, and D_r^i represents the transmission capacity that must be reserved for $r \in R$ on the ith layer. The ILP models are solved with CPLEX Interactive Optimizer 12.8.0.0 [34] on a computer equipped with Intel Xeon CPU E3-1270 and 64 GB of RAM. We set the limit computation time to 1000 seconds.

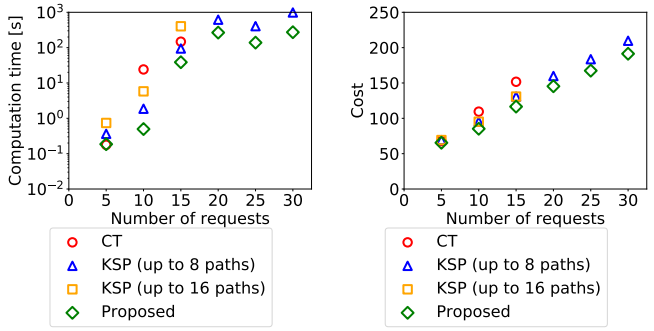
The traffic change rates of VNFs are set so that the set of VNFs includes both traffic-decreasing VNFs and traffic-increasing VNFs along with a VNF that does not change the traffic amount. The maximum transmission capacity of each link, b_l, and the number of CPU cores on each node, c_v, are set to sufficient numbers so that we can obtain a feasible solution. The link utilization cost, Ψ_l, and the VNF placement cost, Ψ_f, are set to the same values to the simulation settings in [8]. The number of CPU cores that each VNF requires per unit of transmission capacity, Δ_f, is set according to the setting in [8], where the processing capacity of each VNF is 10; $\Delta_f = 0.1$ is the reciprocal of 10.

Figs. 4(a) and 4(b) show the computation time and the total cost, respectively, for each case on the 6-node 8-link network. The maximum number of route candidates obtained with the k-shortest path algorithm in the KSP model is set to eight, which is the maximum number of paths that can be found between two different nodes on the 6-node 8-link network. The computation time of the CT model is larger than those of the proposed model and the KSP model. The proposed model cuts the total cost by 33.1% and 0.9% compared to the CT model and the KSP model, respectively, when the number of requests is set to 35. The proposed model cuts the total cost by 0.9% compared to the KSP model when the number of requests is set to 50.



(a) Comparison of computation time. (b) Comparison of total cost.

Fig. 4. Comparison with existing works on 6-node 8-link network.



(a) Comparison of computation time. (b) Comparison of total cost.

Fig. 5. Comparison with existing works on NSFNET.

Figs. 5(a) and 5(b) show the computation time and the total cost, respectively, for each case on NSFNET. The maximum number of route candidates obtained with the k-shortest path algorithm in the KSP model is set to 8 and 16. The proposed model cuts the total cost by 23.2%, 10.9%, and 10.9% compared to the CT model, the KSP model with up to 8 paths, and the KSP model with up to 16 paths, respectively, when the number of requests is set to 15. The proposed model cuts the total cost by 8.8% compared to the KSP model with up to 8 paths when the number of requests is set to 30.

These results show that considering the VNF traffic changes reduces the consumption of network and computation resources compared to the case of assuming constant traffic. In addition, the total cost can be suppressed by allowing looping routes.

Note that the computation time of the ILP problem does not always increase by an increase in the number of requests. CPLEX optimizer uses a branch and cut algorithm [35] when it solves an ILP problem. In the branch and cut algorithm, CPLEX optimizer solves continuous relaxation subproblems that are managed in a tree. The running time of the branch and cut algorithm depends on the number of subproblems that need to be solved and the computation time of each subproblem. The tree of subproblems is extended by branch generations, which are processes that create multiple new subproblems by substituting different integers into a certain variable of a parent subproblem. The number of subproblems exponentially

increases in each branch generation according to the number of variables if there is no additional action to efficiently search subproblems. In the branch and cut algorithm, some constraints, namely cuts, are added to subproblems in order to avoid a solution that includes non-integer variables; cuts generally reduce the necessary number of branches, i.e., the number of subproblems. This means that the computation time does not necessarily increase depending on the size of the problem.

B. Evaluation of order pattern limitation

This section compares the performance of the proposed model achieved when considering all visit order patterns with that when the number of visit order patterns is limited.

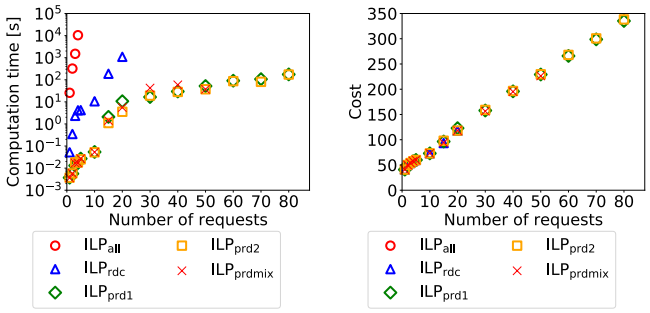
1) Computation time and total cost: We use the 6-node 8-link network shown in Fig. 3(a) and NSFNET. We assume the same set of five VNFs, F, and visit order constraints shown in Section VI-A. Each request randomly chooses at least three and at most five VNFs in F without overlap. Network parameters such as the maximum transmission capacity of each link, the number of CPU cores of each node, the link utilization cost, and the VNF placement cost are set to the values shown in Section VI-A. The maximum transmission capacity of each link and the number of CPU cores on each node are set to sufficient numbers so that we can obtain a feasible solution. The initial transmission capacity of each request is 1.

Let j_1^r and j_2^r be visit order patterns of $r \in R$ obtained with Ma's algorithm by setting $k = 1$ and $k = 2$, respectively. In evaluating the 6-node 8-link network, we compare five cases:

- $\mathrm{ILP}_{\mathrm{all}}$: The ILP model in (15)–(28) where J_r includes all possible order patterns for each request
- $\mathrm{ILP}_{\mathrm{rdc}}$: The ILP model in (15)–(28) where $J_r = \{j_1^r, j_2^r\}$ for each request
- $\mathrm{ILP}_{\mathrm{prd1}}$: The ILP model in (29)–(32) with $J_r = \{j_1^r\}$ for each request
- $\mathrm{ILP}_{\mathrm{prd2}}$: The ILP model in (29)–(32) with $J_r = \{j_2^r\}$ for each request
- $\mathrm{ILP}_{\mathrm{prdmix}}$: The ILP model in (29)–(32) with the visit order obtained by solving the ILP model in (33)–(35) with $J_r = \{j_1^r, j_2^r\}$ for each request

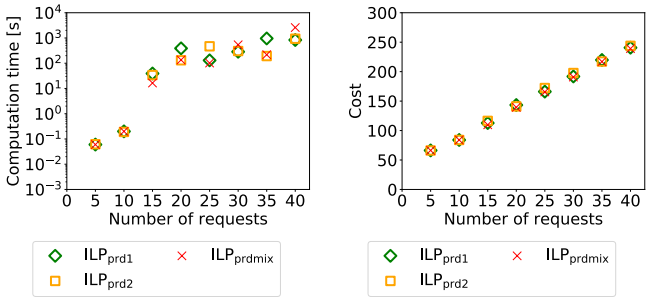
In evaluating NSFNET, we compare three cases: $\mathrm{ILP}_{\mathrm{prd1}}$, $\mathrm{ILP}_{\mathrm{prd2}}$, and $\mathrm{ILP}_{\mathrm{prdmix}}$. These evaluations determine the average values of computation time and total cost in ten scenarios. We set the limit computation time to 1000 seconds. The computation environment is the same as that shown in Section VI-A.

Figs. 6(a) and 6(b) show the computation time and the total cost for each case on the 6-node 8-link network, respectively. The results in $\mathrm{ILP}_{\mathrm{all}}$ when the number of requests is set to 5 or more and those in $\mathrm{ILP}_{\mathrm{rdc}}$ when the number of requests is set to 25 or more are not shown as their computation time exceeded 1000 seconds. The total cost in $\mathrm{ILP}_{\mathrm{all}}$, which considers all possible order patterns, is the optimal total cost in the proposed model. $\mathrm{ILP}_{\mathrm{rdc}}$, $\mathrm{ILP}_{\mathrm{prd1}}$, $\mathrm{ILP}_{\mathrm{prd2}}$, and $\mathrm{ILP}_{\mathrm{prdmix}}$, which limit the number of order patterns, shorten the computation time compared to $\mathrm{ILP}_{\mathrm{all}}$. $\mathrm{ILP}_{\mathrm{prd1}}$, $\mathrm{ILP}_{\mathrm{prd2}}$, and $\mathrm{ILP}_{\mathrm{prdmix}}$, which consider only one order pattern for each request in the ILP model, shorten the computation time compared to $\mathrm{ILP}_{\mathrm{rdc}}$. The total costs in $\mathrm{ILP}_{\mathrm{rdc}}$, $\mathrm{ILP}_{\mathrm{prd1}}$, $\mathrm{ILP}_{\mathrm{prd2}}$, and $\mathrm{ILP}_{\mathrm{prdmix}}$ increase by at most 0.4% compared to $\mathrm{ILP}_{\mathrm{all}}$ when the number of requests is set to 4. The total costs in cases $\mathrm{ILP}_{\mathrm{prd1}}$, $\mathrm{ILP}_{\mathrm{prd2}}$, and $\mathrm{ILP}_{\mathrm{prdmix}}$ increase by 4.4%, 0.2%, and 0.2% compared to $\mathrm{ILP}_{\mathrm{rdc}}$, respectively, when the number of requests is 20. $\mathrm{ILP}_{\mathrm{prdmix}}$ yields less cost than $\mathrm{ILP}_{\mathrm{prd1}}$ and $\mathrm{ILP}_{\mathrm{prd2}}$ when the number of requests is set to 25 or more.



(a) Comparison of computation time. (b) Comparison of total cost.

Fig. 6. Evaluation of order pattern limitation on 6-node 8-link network.



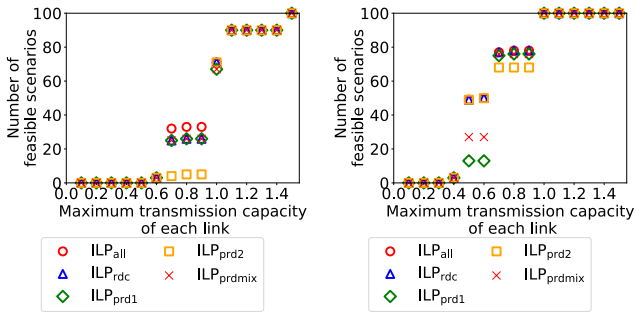
(a) Comparison of computation time. (b) Comparison of total cost.

Fig. 7. Evaluation of order pattern limitation on NSFNET.

Figs. 7(a) and 7(b) show the computation time and the total cost for each case on NSFNET, respectively. The computation time in each NSFNET case is longer than that on the 6-node 8-link network because of the increased number of nodes and links. $\mathrm{ILP}_{\mathrm{prdmix}}$ yields less cost than the other cases when the number of requests is set to 15 or more.

These results show that the computation time can be shortened by limiting the number of order patterns considered in the ILP model, especially when the visit order of each request is predetermined; the penalty is an increase in total cost. Moreover, we can expect to lower the total cost by using the ILP model in (33)–(35) to select the visit order rather than using Ma's algorithm, where k is set to the same value for all requests.

2) Number of feasible scenarios: The limitation in the number of order patterns can cause an inefficient combination of visit orders among requests. Some requests may need to make a detour in order to share the same VNF instance with others; this leads to an increase in the reserved transmission capacity. On the other hand, some requests may avoid sharing VNF instances with others in order to suppress the reserved transmission capacity by shortening the routes or passing



(a) $c_v = 1$. (b) $c_v = 2$.

Fig. 8. Number of feasible scenarios out of 100 scenarios.

through traffic-decreasing VNFs; this leads to an increase in the number of used CPU cores. Therefore, if the maximum transmission capacity of each link and the number of CPU cores on each node are limited, no feasible solution that accommodates all requests on the network can be obtained.

We evaluate the number of feasible scenarios when the number of order patterns is limited. We compare five cases: $\mathrm{ILP}_{\mathrm{all}}$, $\mathrm{ILP}_{\mathrm{rdc}}$, $\mathrm{ILP}_{\mathrm{prd1}}$, $\mathrm{ILP}_{\mathrm{prd2}}$, and $\mathrm{ILP}_{\mathrm{prdmix}}$. 100 scenarios are randomly generated in terms of the source node, the destination node, and required VNFs of each request. Note that the number of feasible scenarios with $\mathrm{ILP}_{\mathrm{all}}$, which considers all possible order patterns, is the maximum number of feasible scenarios in the proposed model. The evaluation is conducted on the 6-node 8-link network where the number of request is two. We vary the maximum transmission capacity of each link, b_l, from 0.1 to 1.5 at an interval of 0.1. We conduct the evaluation in two circumstances where the number of CPU cores on each node, c_v, is set to one and two.

Fig. 8 shows the number of feasible scenarios out of 100 scenarios. Figs. 8(a) and 8(b) show the results when $c_v = 1$ and $c_v = 2$, respectively. There is no feasible scenario when $b_l \leq 0.5, c_v = 1$ or when $b_l \leq 0.3, c_v = 2$. We observe that there are several scenarios where no feasible solution can be obtained if the number of visit order patterns is limited, whereas the optimal solution is obtained with $\mathrm{ILP}_{\mathrm{all}}$ when $0.7 \leq b_l \leq 1.0, c_v = 1$ or $0.4 \leq b_l \leq 0.9, c_v = 2$. For example, when $b_l = 0.7, c_v = 1$, there are 32 feasible scenarios with $\mathrm{ILP}_{\mathrm{all}}$. On the other hand, the numbers of feasible scenarios with $\mathrm{ILP}_{\mathrm{rdc}}$, $\mathrm{ILP}_{\mathrm{prd1}}$, $\mathrm{ILP}_{\mathrm{prd2}}$, and $\mathrm{ILP}_{\mathrm{prdmix}}$ are 25, 25, 4, and 25, respectively. When $b_l = 1.5, c_v = 1$ or $b_l \geq 1.0, c_v = 2$, all of the 100 scenarios are feasible with $\mathrm{ILP}_{\mathrm{all}}$, $\mathrm{ILP}_{\mathrm{rdc}}$, $\mathrm{ILP}_{\mathrm{prd1}}$, $\mathrm{ILP}_{\mathrm{prd2}}$, and $\mathrm{ILP}_{\mathrm{prdmix}}$. In addition, we observe that the numbers of feasible scenarios with $\mathrm{ILP}_{\mathrm{rdc}}$, $\mathrm{ILP}_{\mathrm{prd1}}$, $\mathrm{ILP}_{\mathrm{prd2}}$, and $\mathrm{ILP}_{\mathrm{prdmix}}$ in $c_v = 2$ are larger than those in $c_v = 1$ when $0.4 \leq b_l \leq 1.4$. In particular, when $1.0 \leq b_l \leq 1.4$, all of the 100 scenarios are feasible in $c_v = 2$, while there are at least 10 infeasible scenarios in $c_v = 1$. The results indicate that we can obtain feasible solutions in a shorter time with $\mathrm{ILP}_{\mathrm{rdc}}$, $\mathrm{ILP}_{\mathrm{prd1}}$, $\mathrm{ILP}_{\mathrm{prd2}}$, and $\mathrm{ILP}_{\mathrm{prdmix}}$ than with $\mathrm{ILP}_{\mathrm{all}}$ if the maximum transmission capacity of each link and the number of CPU cores on each node are set to sufficient values.

C. Evaluation of heuristic algorithms

This section compares the performance of the greedy algorithm and the PD algorithm including its variation, PD-TC, with $\mathrm{ILP}_{\mathrm{prd1}}$, $\mathrm{ILP}_{\mathrm{prd2}}$, and $\mathrm{ILP}_{\mathrm{prdmix}}$ shown in Section VI-B. In this evaluation, n_{grd} is set to one and five. The greedy algorithm randomly picks n_{grd} requests from R at line 4 of Algorithm 1. We set the limit computation time to 1000 seconds. We evaluate in three conditions in terms of the visit order constraints. Results shown in this section are obtained with a 95% confidence interval that is not greater than 5% of the average values in terms of the total costs.

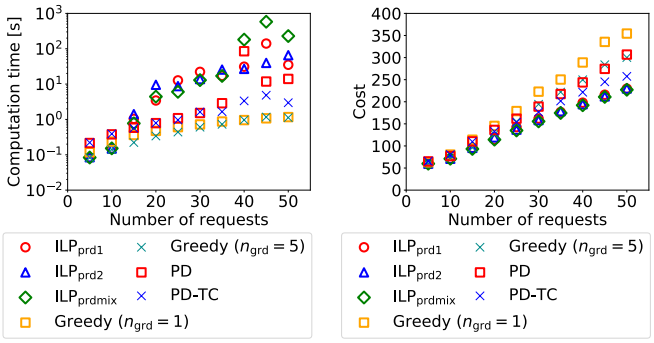
1) Partially-ordered (traffic-decreasing VNFs are required before traffic-increasing ones): We set visit order constraints as follows here:

- f_0 has to be passed through after f_2 if both of them are required.
- f_4 has to be passed through after f_1 if both of them are required.

These constraints are the same with those shown in Section VI-A. Figs. 9(a) and 9(b) show the computation time and the total cost, respectively, on the 6-node 8-link network. Fig. 9 shows the average values in 20 trials. The computation times of the greedy algorithm with $n_{\mathrm{grd}} = 1$, that with $n_{\mathrm{grd}} = 5$, and PD-TC are shorter than those of $\mathrm{ILP}_{\mathrm{prd1}}$, $\mathrm{ILP}_{\mathrm{prd2}}$, and $\mathrm{ILP}_{\mathrm{prdmix}}$ when the number of requests is 15 or more. The computation time of PD exceeds those of $\mathrm{ILP}_{\mathrm{prd1}}$, $\mathrm{ILP}_{\mathrm{prd2}}$, and $\mathrm{ILP}_{\mathrm{prdmix}}$ when the number of requests is 5 and 10, and those of $\mathrm{ILP}_{\mathrm{prd1}}$ and $\mathrm{ILP}_{\mathrm{prd2}}$ when the number of requests is 40. The total costs obtained with the greedy algorithm with $n_{\mathrm{grd}} = 1$, that with $n_{\mathrm{grd}} = 5$, PD, and PD-TC increase by 56.0%, 32.1%, 35.1%, and 13.4% compared to $\mathrm{ILP}_{\mathrm{prdmix}}$, respectively, when the number of requests is 50.

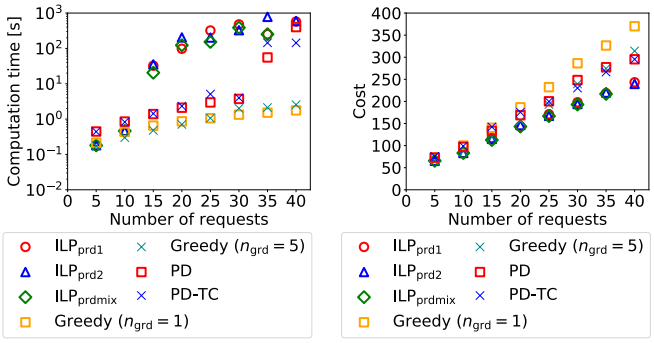
Figs. 10(a) and 10(b) show the computation time and the total cost, respectively, on NSFNET. Fig. 10 shows the average values in 40 trials when the number of requests is 10, in 30 trials when the number of requests is 15, and in 20 trials in other cases. The result of $\mathrm{ILP}_{\mathrm{prdmix}}$ when the number of requests is 40 is not shown since the computation time exceeds 1000 seconds, the limit computation time. The computation times of the greedy algorithm with $n_{\mathrm{grd}} = 1$, that with $n_{\mathrm{grd}} = 5$, PD, and PD-TC are shorter than those of $\mathrm{ILP}_{\mathrm{prd1}}$, $\mathrm{ILP}_{\mathrm{prd2}}$, and $\mathrm{ILP}_{\mathrm{prdmix}}$ when the number of requests is 15 or more. The total costs obtained with the greedy algorithm with $n_{\mathrm{grd}} = 1$, that with $n_{\mathrm{grd}} = 5$, PD, and PD-TC increase by 54.5%, 31.1%, 23.3%, and 23.3% compared to $\mathrm{ILP}_{\mathrm{prd2}}$, respectively, when the number of requests is 40.

We observe that the total cost obtained with the greedy algorithm with $n_{\mathrm{grd}} = 1$ is the largest when the number of requests is 10 or more on both the 6-node 8-link network and NSFNET. This is because the greedy algorithm with $n_{\mathrm{grd}} = 1$ determines request routes and VNF placement with only one request for each iteration and does not consider the remaining requests. This leads to inefficiency in terms of request routes and VNF placement. The total cost is expected to decrease as n_{grd} increases since more requests are considered in the ILP model in (29)–(32) at a time. In fact, the total cost obtained



(a) Comparison of computation time. (b) Comparison of total cost.

Fig. 9. Evaluation of heuristic algorithms on 6-node 8-link network (traffic-decreasing VNFs are required before traffic-increasing ones).



(a) Comparison of computation time. (b) Comparison of total cost.

Fig. 10. Evaluation of heuristic algorithms on NSFNET (traffic-decreasing VNFs are required before traffic-increasing ones).

with the greedy algorithm with $n_{\mathrm{grd}} = 5$ is always smaller than that with $n_{\mathrm{grd}} = 1$.

We also observe that the total cost obtained with PD is larger than that obtained with PD-TC when the number of requests is 15 or more on the 6-node 8-link network and when the number of requests is 25 or more on NSFNET. This is because PD tends to place VNFs on nodes those are in the middle between the source and the destination of a request. The number of hops between the source and the node on which the first VNF is placed can be larger than that with PD-TC. Similarly, the number of hops between the destination and the node on which the final VNF is placed can be larger than that with PD-TC. These behaviors of PD can cause an increase in the total cost since the sum of reserved link capacity can increase.

2) Partially-ordered (traffic-increasing VNFs are required before traffic-decreasing ones): We set visit order constraints as follows here:

- f_0 has to be passed through after f_3 if both of them are required.
- f_1 has to be passed through after f_4 if both of them are required.

Figs. 11(a) and 11(b) show the computation time and the total cost, respectively, on the 6-node 8-link network. Fig. 11 shows the average values in 30 trials when the number of requests is 10 and in 20 trials in other cases. The computation times of the greedy algorithm with $n_{\mathrm{grd}} = 1$, that with $n_{\mathrm{grd}} = 5$, PD, and PD-TC are shorter than those of $\mathrm{ILP}_{\mathrm{prd1}}$, $\mathrm{ILP}_{\mathrm{prd2}}$, and $\mathrm{ILP}_{\mathrm{prdmix}}$ when the number of requests is 10 or more. The total costs obtained with the greedy algorithm with $n_{\mathrm{grd}} = 1$, that with $n_{\mathrm{grd}} = 5$, PD, and PD-TC increase by 37.7%, 21.2%, 38.0% and 15.1% compared to $\mathrm{ILP}_{\mathrm{prdmix}}$, respectively, when the number of requests is 50.

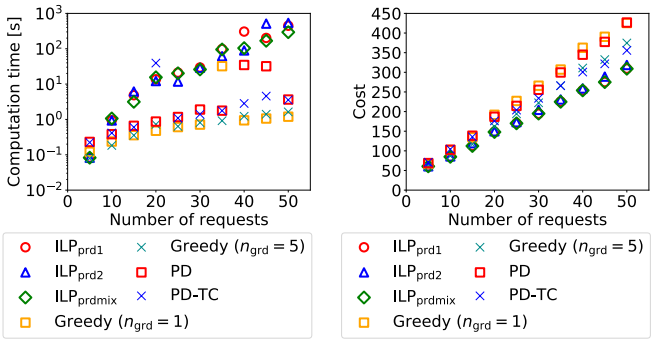
Figs. 12(a) and 12(b) show the computation time and the total cost, respectively, on NSFNET. Fig. 12 shows the average values in 30 trials when the number of requests is 5 and 10 and in 20 trials in other cases. The results of $\mathrm{ILP}_{\mathrm{prd1}}$, $\mathrm{ILP}_{\mathrm{prd2}}$, and $\mathrm{ILP}_{\mathrm{prdmix}}$ when the number of requests is 30 or more are not shown since the computation time exceeds 1000 seconds, the limit computation time. The computation times of the greedy algorithm with $n_{\mathrm{grd}} = 1$, that with $n_{\mathrm{grd}} = 5$, PD, and PD-TC are shorter than those of $\mathrm{ILP}_{\mathrm{prd1}}$, $\mathrm{ILP}_{\mathrm{prd2}}$, and $\mathrm{ILP}_{\mathrm{prdmix}}$ when the number of requests is 10 or more. The total costs obtained with the greedy algorithm with $n_{\mathrm{grd}} = 1$, that with $n_{\mathrm{grd}} = 5$, PD, and PD-TC increase by 36.1%, 16.9%, 31.1% and 33.4% compared to $\mathrm{ILP}_{\mathrm{prdmix}}$, respectively, when the number of requests is 25.

We observe that the total cost obtained with the greedy algorithm with $n_{\mathrm{grd}} = 5$ is always smaller than that obtained with PD and PD-TC on NSFNET, which is different from the results shown in Sections VI-C1 and VI-C3. In the case applying the greedy algorithm with $n_{\mathrm{grd}} = 5$, VNFs tend to be concentratedly placed on fewer nodes than the cases with PD and PD-TC. Some requests pass through all of their required VNFs on a particular node. On the other hand, in the cases with PD and PD-TC, VNFs tend to be distributed since PD and PD-TC place VNFs on nodes near the source or the destination of requests. This can cause roundabout and round-trip routes. When we set visit order constraints such that a traffic-increasing VNF is required before a traffic-decreasing one, the traffic increased by the former VNF is forwarded to the latter VNF. If the two VNFs are placed on different nodes, links between the nodes need to reserve transmission capacity for the increased traffic. These aspects lead to increases in the total costs obtained with PD and PD-TC.

The difference in total costs between PD and PD-TC varies depending on the network size. The total cost obtained with PD is larger than that with PD-TC by 14.7% on the 6-node 8-link network when the number of requests is 40. On the other hand, the total cost obtained with PD is larger than that with PD-TC by 2.2% on NSFNET when the number of requests is 40. These results indicate that the relative difference in total costs between PD and PD-TC on NSFNET is smaller than that on the 6-node 8-link network. This is because the sources or the destinations of requests can be distributed and be distant from each other on a larger network. This leads to an increase in the number of hops between the source and the node on which VNFs are placed. Similarly, the number of hops between the destination and the node on which VNFs are placed can increase.

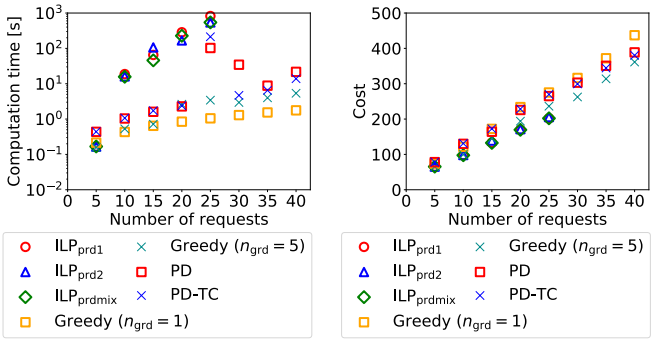
3) Non-ordered: We set no visit order constraint here.

Figs. 13(a) and 13(b) show the computation time and the total cost, respectively, on the 6-node 8-link network. Fig. 13 shows the average values in 20 trials. We do not show the results in $\mathrm{ILP}_{\mathrm{prd2}}$ and $\mathrm{ILP}_{\mathrm{prdmix}}$ since the visit orders in these



(a) Comparison of computation time. (b) Comparison of total cost.

Fig. 11. Evaluation of heuristic algorithms on 6-node 8-link network (traffic-increasing VNFs are required before traffic-decreasing ones).



(a) Comparison of computation time. (b) Comparison of total cost.

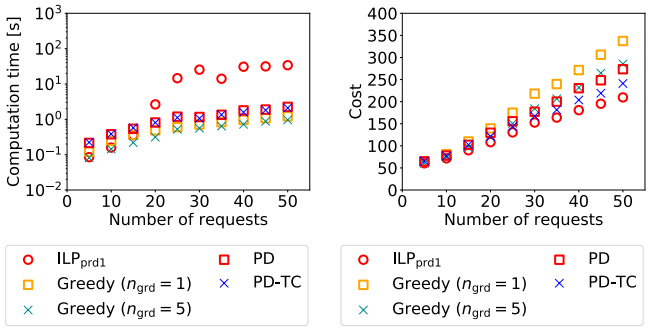
Fig. 12. Evaluation of heuristic algorithms on NSFNET (traffic-increasing VNFs are required before traffic-decreasing ones).

cases are the same to that in ILP$_{\mathrm{prd1}}$; the total costs are also the same. The computation times of the greedy algorithm with $n_{\mathrm{grd}} = 1$, that with $n_{\mathrm{grd}} = 5$, PD, and PD-TC are shorter than that of ILP$_{\mathrm{prd1}}$ when the number of requests is 20 or more. The total costs obtained with the greedy algorithm with $n_{\mathrm{grd}} = 1$, that with $n_{\mathrm{grd}} = 5$, PD, and PD-TC increase by 60.9%, 36.1%, 30.5%, and 15.0% compared to ILP$_{\mathrm{prd1}}$, respectively, when the number of requests is 50.

Figs. 14(a) and 14(b) show the computation time and the total cost, respectively, on NSFNET. Fig. 14 shows the average values in 30 trials when the number of requests is 20 and in 20 trials in other cases. The computation times of the greedy algorithm with $n_{\mathrm{grd}} = 1$, that with $n_{\mathrm{grd}} = 5$, PD, and PD-TC are shorter than that of ILP$_{\mathrm{prd1}}$ when the number of requests is 15 or more. The total costs obtained with the greedy algorithm with $n_{\mathrm{grd}} = 1$, that with $n_{\mathrm{grd}} = 5$, PD, and PD-TC increase by 55.1%, 32.3%, 24.3%, and 17.3% compared to ILP$_{\mathrm{prd1}}$, respectively, when the number of requests is 40.

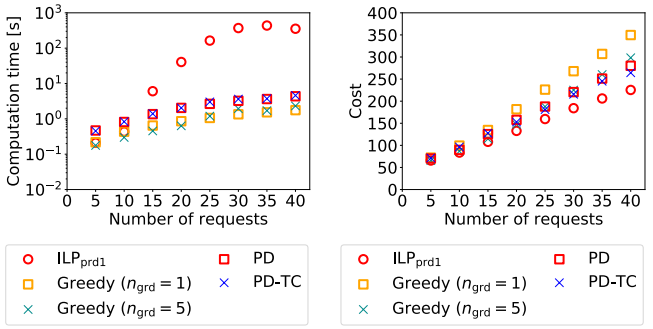
D. Evaluation in realistic situations

This section compares the performances of the greedy algorithm with $n_{\mathrm{grd}} = 5$ and PD-TC in realistic situations. In this evaluation, we use five types of service chains shown in Table V. The demand shown in Table V is the ratio of the traffic amount of a service to the total traffic amount. For example, the sum of the traffic of the web service is 120 GB when the total traffic amount is 1 TB. The number



(a) Comparison of computation time. (b) Comparison of total cost.

Fig. 13. Evaluation of heuristic algorithms on 6-node 8-link network (non-ordered).



(a) Comparison of computation time. (b) Comparison of total cost.

Fig. 14. Evaluation of heuristic algorithms on NSFNET (non-ordered).

of requests for each service is determined by solving the following simultaneous equations:

$$\sum_{s\in S} x_s = |R|, \tag{44}$$

$$\sum_{s\in S} D_s x_s = D_{\mathrm{total}}, \tag{45}$$

$$D_s x_s = P_s D_{\mathrm{total}}, \forall s \in S. \tag{46}$$

S is a set of service types. x_s is a variable that represents the number of requests for service $s \in S$. Note that x_s can be a non-integer number. D_s is the transmission capacity of $s \in S$. P_s is the demand of $s \in S$. D_{total} is a variable that represents the total traffic amount. Equation (44) represents that the sum of the number of requests for each service is $|R|$, the number of requests. Equation (45) represents that the sum of the traffic amount of each service is the total traffic amount. Equation (46) represents that the traffic amount of $s \in S$ is the product of the demand of s and the total traffic amount. After the simultaneous equations are solved, we round x_s down to the nearest integer x'_s. If $\sum_{s\in S} x'_s$ is less than $|R|$, we randomly select one service s' and increment $x'_{s'}$. We iterate this until $\sum_{s\in S} x'_s$ becomes $|R|$.

Table VI shows the number of required CPU cores and the throughput of VNFs. We assume that a NAT VNF and a firewall (FW) VNF are combined into one VNF according to Juniper vSRX virtual firewall, which also includes NAT function [26], [28]. We call this combined VNF a NAT-FW. Note that a file sharing request is assumed to pass through a

FW VNF and a NAT VNF independently. In this evaluation, we assume that a FW for a file sharing is executed by a NAT-FW which can be shared with other services. On the other hand, we assume that a NAT VNF for a file sharing cannot be shared with other services. We refer to a NAT VNF for a file sharing as NAT-FW2, which is identical with a NAT-FW except that a NAT-FW2 is used by only file sharing requests. Δ_f, the number of CPU cores that VNF f requires, can be determined by dividing the number of required CPU cores by the throughput. For example, $\Delta_{\text{NAT-FW}} = 1/\text{Gbps}$ since the number of CPU cores that a NAT-FW requires is two and the throughput of a NAT-FW is 2 Gbps.

The traffic change rate of a WAN optimization controller (WOC) VNF is set to 0.2 according to the Citrix NetScaler SD-WAN WAN optimizer, which reduces the traffic amount by 80% [36]. The traffic change rate of a video optimization controller (VOC) VNF is set to 0.5 according to Akamai Image and Video Manager, which reduces the file size by 50% when it converts the video from MP4 (H.264) to MP4 (H.265) or WebM (V9) [37]. The traffic change rate of a traffic monitor (TM) VNF is set to 1.0 since it does not affect the transmission capacity of a request. A firewall VNF, an intrusion detection prevention system (IDPS) VNF, and a deep packet inspection (DPI) VNF discard the packets against the security policies of the VNFs. Therefore, the traffic change rates of a NAT-FW, an IDPS VNF, and a DPI VNF is 1.0 for the admitted traffic and 0 for the rejected traffic. In this evaluation, we fix the traffic change rates of a NAT-FW, an IDPS VNF, and a DPI VNF to 1.0 as with the experiment conducted in [4], that is, we assume that all packets are admitted. A traffic shaper (TS) VNF buffers the data if the traffic amount exceeds the threshold. The buffered data are sent when the traffic amount decreases below the threshold. If there is no remaining buffer space, the excessive data are discarded. The transmission capacity required to be reserved for the service can be reduced by using a TS VNF. In this evaluation, we assume that a TS VNF suppresses the required transmission capacity of the egress flow to 80% of that of the ingress flow; the traffic change rate of a TS VNF is set to 0.8.

Visit order constraints are set as follows:

- A WOC VNF has to be passed through after a NAT-FW if both of them are required.
- An IDPS VNF has to be passed through after a WOC VNF if both of them are required.
- An IDPS VNF has to be passed through after a VOC VNF if both of them are required.
- A TS VNF has to be passed through after a DPI VNF if both of them are required.
- A NAT-FW2 has to be passed through after an IDPS VNF if both of them are required.

We vary the number of requests, $|R|$, from 100 to 1000 at an interval of 100. Note that, in the case where all five services are provisioned for all source-destination pairs on NSFNET, the number of requests is $14 \times 13 \times 5 = 910$; we set the maximum number of $|R|$ to 1000 so that it is larger than the number of possible request patterns.

The maximum transmission capacity of each link is 10 Gbps, i.e., $b_l = 10\,\text{Gbps}$. The number of CPU cores attached to each node is 20, i.e., $c_v = 20$. Similar to the evaluations in Sections VI-A–VI-C, the maximum transmission capacity of each link and the number of CPU cores on each node are set to sufficient numbers so that we can obtain a feasible solution. The link utilization cost and the VNF placement cost depend on how network service providers configure the network system. Therefore, we evaluate the performances of the greedy algorithm and PD-TC by varying the values of Ψ_l, the link utilization cost of each link, and Ψ_f, the VNF placement cost of each VNF. We use NSFNET in this section. The results shown in this section are obtained with a 95% confidence interval that is not greater than 5% of the average values in terms of the total costs.

First, we set $\Psi_l = 1/\text{Mbps}$ and $\Psi_f = 10$. Figs. 15(a) and 15(b) show the computation time and the total cost, respectively. Fig. 15 shows the average values in 70 trials when $|R| = 100$, in 60 trials when the number of requests is $|R| = 200$, in 50 trials when $300 \leq |R| \leq 600$, and in 40 trials in other cases. The computation time of PD-TC is shorter than that of the greedy algorithm by 88.1% when $|R| = 1000$. The total cost obtained with PD-TC is smaller than that with the greedy algorithm by 2.8% when $|R| = 1000$. The proportions of the link utilization cost to the total cost obtained with the greedy algorithm and PD-TC are 89.8% and 91.8%, respectively, when $|R| = 1000$. In the rest of this section, we discuss the numerical results by using these results as the baseline values.

Next, we set $\Psi_l = 10/\text{Mbps}$ and $\Psi_f = 10$; Ψ_l is ten times larger than that in the case shown in Fig. 15 for all links in L. Figs. 16(a) and 16(b) show the computation time and the total cost, respectively. Fig. 16 shows the average values in 90 trials when $|R| = 100$, in 70 trials when $|R| = 100$, in 60 trials when $300 \leq |R| \leq 600$, in 50 trials when $|R| = 700$, and in 40 trials in other cases. The total cost obtained with the greedy algorithm and PD-TC are 4.6 times and 9.3 times, respectively, larger than the case of Fig. 15 when $|R| = 1000$. The proportion of the link utilization cost to the total cost obtained with the greedy algorithm is larger by 0.6 percentage point than the case of Fig. 15 when $|R| = 1000$. On the other hand, the proportion of the link utilization cost to the total cost obtained with PD-TC is larger by 7.3 percentage point than the case of Fig. 15. The reason is explained as follows. The greedy algorithm determines VNF placement and request routes at a time for each n_{grd} requests. When Ψ_l increases, the greedy algorithm tends to deploy more VNF instances so that the total number of hops, and thus, the link utilization cost can be suppressed. For example, the link utilization cost is 4.7 times but not 10 times larger than the case of Fig. 15 when $|R| = 1000$; this indicates that the link utilization cost is suppressed with the increase in the CPU utilization cost. On the other hand, PD-TC determines request routes after VNF placement is obtained in each iteration. Furthermore, PD-TC tends to deploy VNFs so that the total number of used CPU cores is as small as possible since the while loop of determining VNF placement and request routes in PD-TC starts from the minimum number of CPU cores obtained with the ILP problem in (33)–(35). This leads to increase in the number of hops between the source and the node where traffic-

TABLE V
SERVICE CHAINS (EVALUATION IN REALISTIC SITUATIONS) [6], [12], [29], [30].

Service	Required VNFs	Transmission capacity	Demand
Web service	NAT-FW-TM-WOC-IDPS	100 kbps	12%
VoIP	NAT-FW-TM-FW-NAT	64 kbps	11%
Video streaming	NAT-FW-TM-VOC-IDPS	4 Mbps	71%
Online gaming	NAT-FW-VOC-WOC-IDPS	4 Mbps	4%
File sharing	DPI-TS-FW-IDPS-NAT	5 Mbps	2%

TABLE VI
NUMBER OF REQUIRED CPU CORES AND THROUGHPUT OF VNFS [26]–[28], [31], [32].

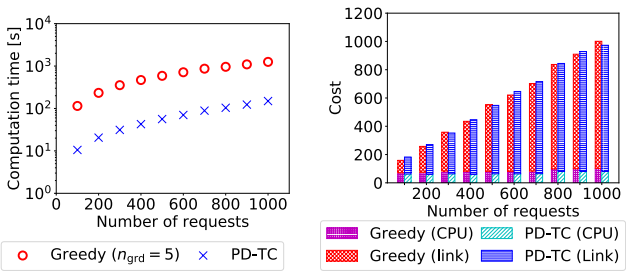
VNF	Number of CPU cores	Throughput
NAT-FW	2	2 Gbps
TM	1	1 Gbps
WOC	1	0.5 Gbps
IDPS	4	1 Gbps
VOC	2	2 Gbps
DPI	1	10 Gbps
TS	1	1 Gbps

decreasing VNFs are placed, and thus, leads to increase in the link utilization cost. For example, the link utilization cost is 10 times larger than the case of Fig. 15 when $|R| = 1000$; this corresponds to that Ψ_l is 10 times larger.

Finally, we set $\Psi_l = 1/\text{Mbps}$ and $\Psi_f = 100$; Ψ_f is ten times larger than that in the case shown in Fig. 15 for all VNFs in F. Figs. 17(a) and 17(b) show the computation time and the total cost, respectively. Fig. 17 shows the average values in 80 trials when $|R| = 100$, in 60 trials when $|R| = 200$, in 50 trials when $300 \leq |R| \leq 600$, and in 40 trials in other cases. The total cost obtained with the greedy algorithm and PD-TC are 2.0 times and 1.7 times, respectively, larger than the case of Fig. 15 when $|R| = 1000$. The proportions of the link utilization cost to the total cost obtained with the greedy algorithm and PD-TC are smaller by 30.3 and 39.1 percentage points, respectively, than the case of Fig. 15 when $|R| = 1000$. The reason is explained as follows. Both the greedy algorithm and PD-TC tend to deploy VNFs so that the total number of used CPU cores is as small as possible when Ψ_f increases. Indeed, the number of used CPU cores obtained with the greedy algorithm is the same with that obtained with PD-TC; the difference in the total cost comes only from the difference in the link utilization cost. The greedy algorithm first determines the optimal VNF placement for the first n_{grd} requests. The rest of the requests tend to pass through the VNF instances which the first n_{grd} requests pass through in order not to increase the number of used CPU cores. This can cause inefficiency in the link utilization for the requests other than the first n_{grd} requests. On the other hand, PD-TC determines the VNF placement for all requests at a time so that the traffic-decreasing VNFs are placed on the nodes near to the request sources.

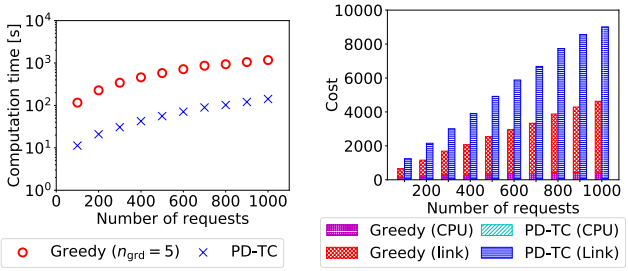
E. Large-size networks

We additionally conduct the evaluation by using three large-size networks, namely BICS, Viatel, and US Carrier [38],



(a) Comparison of computation time. (b) Comparison of total cost.

Fig. 15. Evaluation in realistic situations (NSFNET, $\Psi_l = 1/\text{Mbps}$, $\Psi_f = 10$).



(a) Comparison of computation time. (b) Comparison of total cost.

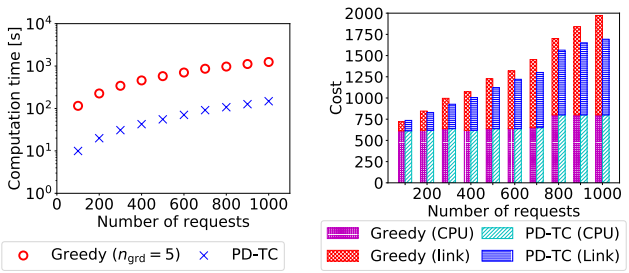
Fig. 16. Evaluation in realistic situations (NSFNET, $\Psi_l = 10/\text{Mbps}$, $\Psi_f = 10$).

which are shown in Figs. 18, 19, and 20, respectively. The evaluation settings other than the network topology are the same as those in Section VI-D.

Fig. 21 shows the computation time and the total cost in $\Psi_l = 1/\text{Mbps}$ and $\Psi_f = 10$ on BICS. The computation time with the greedy algorithm on BICS is 7.0 times larger than that on NSFNET when $|R| = 200$. The computation time with PD-TC on BICS is 2.3 times larger than that on NSFNET when $|R| = 1000$. Compared to the results on NSFNET, the computation time with both algorithms increases due to the increase of network size. The total cost with the greedy algorithm on BICS is 1.2 times larger than that on NSFNET when $|R| = 200$. Specifically, the link utilization cost and the CPU utilization cost with the greedy algorithm on BICS are 1.3 and 1.2 times, respectively, larger than those on NSFNET; both link utilization cost and CPU utilization cost have an impact on the increase in the total cost. On the other hand, the total cost with PD-TC on BICS is 1.3 times larger than that on NSFNET when $|R| = 1000$. The CPU utilization cost with PD-TC on BICS when $|R| = 1000$ is the same as that on NSFNET; the increase in the total cost is due to the increase in the link utilization cost.

Fig. 22 shows the computation time and the total cost in $\Psi_l = 1/\text{Mbps}$ and $\Psi_f = 10$ on Viatel. The results with the greedy algorithm are not shown since the computation time exceeds 1000 seconds. The computation time with PD-TC on Viatel is 6.6 times larger than that on NSFNET when $|R| = 1000$. The total cost with PD-TC on Viatel is 4.8 times larger than that on NSFNET when $|R| = 1000$.

Fig. 23 shows the computation time and the total cost in



(a) Comparison of computation time.

(b) Comparison of total cost.

Fig. 17. Evaluation in realistic situations (NSFNET, $\Psi_l = 1/\text{Mbps}$, $\Psi_f = 100$).

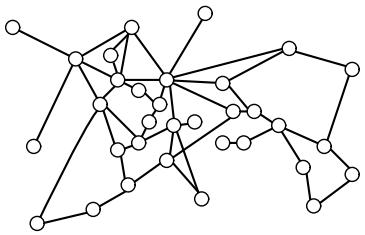


Fig. 18. BICS.

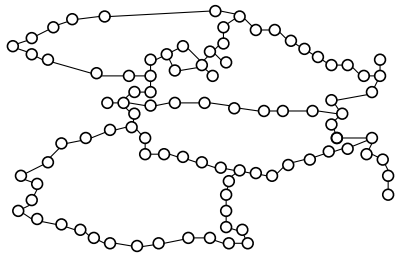


Fig. 19. Viatel.

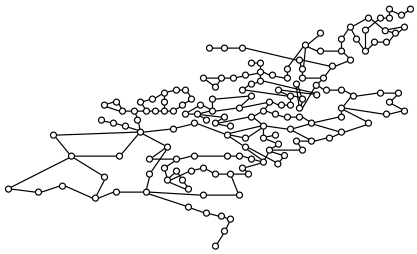


Fig. 20. US Carrier.

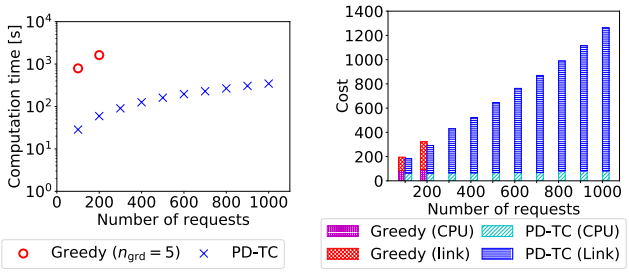
$\Psi_l = 1/\text{Mbps}$ and $\Psi_f = 10$ on US Carrier. The results with the greedy algorithm are not shown since the computation time exceeds 1000 seconds. The computation time with PD-TC on US Carrier is 15.7 times larger than that on NSFNET when $|R| = 600$. The total cost with PD-TC on US Carrier is 3.2 times larger than that on NSFNET when $|R| = 600$. The total cost with PD-TC on US Carrier is smaller by 10.2% than that on Viatel when $|R| = 600$, whereas the number of nodes and links on US Carrier are 1.7 and 2.0 times, respectively, larger than those on Viatel. This is because the average number of hops between two nodes on US Carrier is smaller than that on Viatel; those on US Carrier and Viatel are 12.1 and 13.1, respectively. The increase in the number of hops leads to the increase in the link utilization cost. Note that the CPU utilization costs on US Carrier and Viatel are the same with each other when $|R| = 600$.

VII. Conclusion

This paper proposed a service chain provisioning model that considers the traffic changes created by VNFs while determining the VNF visit order of each request, route, and VNF placement flexibly. The proposed model expresses the traffic amounts changed by VNFs by using a logical layered network model, which also relaxes routing constraints. When the traffic amounts are given as decision variables and parameters of visit order constraints are introduced to the optimization problem, the objective function and some constraints become non-linear. In order to formulate service chain provisioning as an ILP problem, the traffic amount on each layer is given as constants by calculating the values in advance. It is difficult to obtain the optimal solution in practical time if we consider all possible visit order patterns. Therefore, we introduced three methods of limiting the number of visit order patterns so as to shorten the computation time; the methods use Ma's algorithm. In order to handle a problem that is intractable with the ILP model, we introduced a greedy algorithm and an algorithm that divides the problem into the VNF placement part and the routing part. Numerical results showed that the total cost can be reduced by 34.4% when considering VNF traffic changes than when assuming constant traffic amounts. The results also showed that the computation time can be shortened by limiting the number of order patterns considered in the ILP model, especially when the visit order of each request is predetermined, with at most 0.4% of increase in the total cost. We observed that one of the heuristic algorithms, PD-TC, decreases the computation time with at most 23.3% of increase in the total cost compared to the case where we solve the ILP model.

Appendix A
Number of Decision Variables with Augmented Network

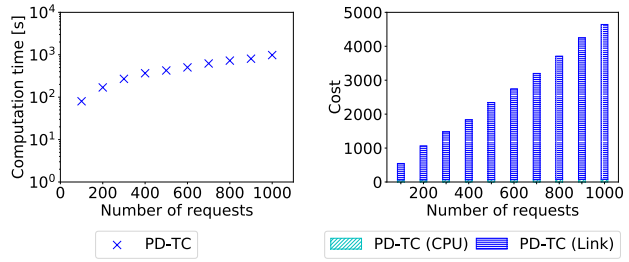
Sasabe *et al.* [12] presented an augmented network, which allows request routes to make loops similarly to a logical layered network. An augmented network has imaginary nodes that correspond to required VNFs. For example, imaginary nodes v_f and v_g in Fig. 2(c) correspond to VNFs f and g, respectively. A virtual link connects a physical node and an imaginary node. Each imaginary node is connected to all physical nodes by virtual links. An augmented network allows looping routes by considering subpaths. The service path of request r on an augmented network consists of n_r+1 subpaths, where n_r is the number of required VNFs. The source of the first subpath is the source of the request. The destination of the $(n_r + 1)$th subpath is the destination of the request. The destination of the ith subpath and the source of the ith subpath is imaginary node $v_{f_i^r}$, where f_i^r is the ith VNF of r. The ingress flow of the ith subpath to $v_{f_i^r}$ and the egress flow of



(a) Comparison of computation time.

(b) Comparison of total cost.

Fig. 21. Evaluation in realistic situations (BICS, $\Psi_l = 1/\text{Mbps}$, $\Psi_f = 10$).



(a) Comparison of computation time.

(b) Comparison of total cost.

Fig. 22. Evaluation in realistic situations (Viatel, $\Psi_l = 1/\text{Mbps}$, $\Psi_f = 10$).

the $(i+1)$th subpath from $v_{f_i^r}$ must pass through the same virtual link connected to $v_{f_i^r}$.

We discuss the number of decision variables when an logical layered network or an augmented network is adopted to the proposed model. Let G_{aug} represent an augmented network. V^+ is a set of nodes in G_{aug}. L_{aug} is a set of links in G_{aug}. $v_{\text{s}}^{r,i}$ and $v_{\text{d}}^{r,i}$ are the source and the destination of the ith subpath for r, respectively. v_f is the imaginary node corresponding to $f \in F$. f_i^r is the ith VNF for r. An augmented network based ILP model for requests whose visit orders are predetermined is formulated as follows:

$$\text{minimize} \quad \sum_{r\in R}\sum_{l\in L}\sum_{i=0}^{n_r} D_r^i \phi_l^{r,i} \Psi_l + \sum_{v\in V}\sum_{f\in F} A_v^f \Psi_f, \tag{47}$$

$$\sum_{l\in\omega^+(v_{\text{s}}^{r,i})} \phi_l^{r,i} = 1, \forall r \in R, 0 \le i \le n_r, \tag{48}$$

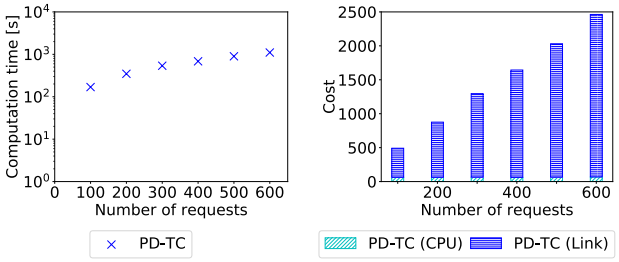
$$\sum_{l\in\omega^-(v_{\text{d}}^{r,i})} \phi_l^{r,i} = 1, \forall r \in R, 0 \le i \le n_r, \tag{49}$$

$$\sum_{l\in\omega^+(v)} \phi_l^{r,i} = \sum_{l\in\omega^-(v)} \phi_l^{r,i},$$
$$\forall v \in V^+ \setminus \{v_{\text{s}}^{r,i}, v_{\text{d}}^{r,i}\}, r \in R, 0 \le i \le n_r, \tag{50}$$

$$\phi_{l_{u,v_{f_i^r}}}^{r,i} = \phi_{l_{v_{f_i^r},u}}^{r,i+1},$$
$$\forall l_{u,v_{f_i^r}} \in L_{\text{aug}}, l_{v_{f_i^r},u} \in L_{\text{aug}}, r \in R, 0 \le i < n_r, \tag{51}$$

$$\phi_{l_{u,v_{f_r^m}}}^{r,i} = 0,$$
$$\forall l_{u,v_{f_r^m}} \in L_{\text{aug}}, r \in R, 0 \le i \le n_r, m \ne i, \tag{52}$$

$$\phi_{l_{v_f,u}}^{r,0} = 0,$$



(a) Comparison of computation time.

(b) Comparison of total cost.

Fig. 23. Evaluation in realistic situations (US Carrier, $\Psi_l = 1/\text{Mbps}$, $\Psi_f = 10$).

$$\forall l_{v_f,u} \in L_{\text{aug}}, f \in F, r \in R, \tag{53}$$

$$\sum_{r\in R}\sum_{i=0}^{n_r} D_r^i \phi_l^{r,i} \le b_l, \forall l \in L, \tag{54}$$

$$\sum_{f\in F} A_v^f \le c_v, \forall v \in V, \tag{55}$$

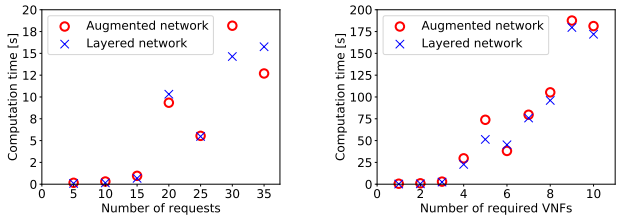
$$\sum_{r\in R}\sum_{i=0}^{n_r-1} \Delta_f D_r^i e_f^{r,i} \phi_{l_{v,v_{f_i^r}}}^{r,i} \le A_v^f, \forall f \in F, v \in V. \tag{56}$$

The number of constraints in (47)–(56) is $|V|((|F|+1)\sum_{r\in R} n_r + 2|R||F| + |R| + |F| + 1) + |L| + \sum_{r\in R} n_r$. This is larger than that in (29)–(32), which adopts a logical layered network, by $|F||V|\sum_{r\in R}(n_r+2) + 2\sum_{r\in R} n_r$. The number of decision variables in (47)–(56) is $|L|\sum_{r\in R}(n_r+1) + |F||V|(\sum_{r\in R} 2(n_r+1)+1)$. This is larger than that in (29)–(32) by $|V|(2|F|-1)\sum_{r\in R}(n_r+1) + |F||V||R|$. For example, in the case where $|F| = 5, |R| = 50, n_r = 5$ for all $r \in R$, and the network topology is the 6-node 8-link network, the number of constraints in (47)–(56) and that in (29)–(32) are 12852 and 1852, respectively. Similarly, the number of decision variables in (47)–(56) and that in (29)–(32) are 22830 and 6330, respectively.

Fig. 24(a) shows the impact of the number of requests on computation time when each network is adopted to the proposed model. The number of required VNFs is at least three and at most five for each request. Fig. 24(b) shows the impact of the number of required VNFs on computation time. Note that the number of layers in a logical layered network is one more than the number of required VNFs. The number of requests is 30. Comparison of the total cost is not shown since the total cost is always the same in each case.

Appendix B
Ma's Algorithm

Dependency relations among VNFs are represented in the form of trees. We determine the visit order by removing vertices from the trees according to the algorithm. We treat VNF f as independent if there is no VNF that needs to be passed through before VNF f, and we treat VNF g as dependent on VNF f if VNF g must be passed through after VNF f.



(a) Impact of number of requests. (b) Impact of number of required VNFs.

Fig. 24. Comparison of computation time (evaluation of augmented network).

F_r represents the set of VNFs that request r requires. O_r represents a totally-ordered set of VNFs expressing the visit order of r. Let O_r initially be an empty set. Request r needs to pass through VNFs in the order in which VNFs are added to O_r. Trees whose root is an independent VNF and VNFs depending on VNF f are regarded as children of VNF f. $k_{\max}$ is the maximum size of the trees, where the size of a tree equals to the number of VNFs that the tree contains.

The procedure of Ma's algorithm is as follows. First, we determine the value of parameter k. k is an integer from 1 to $k_{\max}$. Second, we calculate trees of size up to k whose root is an independent VNF and the product of traffic change rates of all VNFs contained in the tree is minimum. Let T_r denote the set of trees in this phase. The algorithm selects the tree with the minimum product of traffic change rates from T_r, removes its root, namely VNF g, from the tree, and adds VNF g to O_r. The algorithm calculates trees whose root is VNF g', which depends on VNF g, and the product of traffic change rates of all VNFs contained in the tree is minimum. The algorithm then adds these trees to T_r. After this, the algorithm iteratively removes VNFs and adds them to O_r until all required VNFs are added to O_r.

Appendix C
Hardness Analyses of Problems Solved in PD Algorithm

We analyze the hardness of VPP-PD, which is formulated in the ILP problem in (36)–(39). We define the decision version of VPP-PD (VPP-PD-D) as follows:

Definition 2. Bidirected graph $G = (V, L)$, set of requests R, and set of VNFs F are given. Logical layered network G^{L} is constructed from $G = (V, L)$. The number of layers in G^{L} is $n_{\max} = \max_{r \in R} n_r$, where n_r is the number of required VNFs for $r \in R$. The total number of CPU cores on $v \in V$ is c_v. The source and destination nodes of $r \in R$ are $v_{\mathrm{s}}^r \in V$ and $v_{\mathrm{d}}^r \in V$, respectively. The transmission capacity of $r \in R$ on the ith layer of G^{L} is D_r^i. $f \in F$ requires Δ_f CPU cores per unit of transmission capacity. Let v_i^r be the node where the ith VNF of $r \in R$ is deployed. We define $d_{\mathrm{sum}} = \sum_{r \in R} \sum_{v \in V} \sum_{i=0}^{n_r - 1} (d(v_{\mathrm{s}}^r, v_i^r) + d(v_{\mathrm{d}}^r, v_i^r))$, where $d(v_{\mathrm{s}}^r, v_i^r)$ and $d(v_{\mathrm{d}}^r, v_i^r)$ represent the distance between the source and v_i^r and that between the destination and v_i^r, respectively. Is there any VNF placement such that d_{sum} is T or less?

Theorem 2. VPP-PD-D is NP-complete.

Proof. First, we show that VPP-PD-D is in NP. The time complexities of confirming whether each constraint is satisfied are as follows:

- Node capacity: $O(|V||F|)$
- Determining required number of CPU cores for each VNF on each node: $O(n_{\max}|R||V||F|)$
- Selecting node for each VNF of each request: $O(|R||V|)$
- Maximum number of used CPU cores: $O(|V||F|)$

The time complexity of checking whether d_{sum} is T or less is $O(n_{\max}|R||V|)$. Therefore, VPP-PD-D is in NP.

Second, we prove that BPP-D, which is NP-complete, is reducible to VPP-PD-D. We construct a VPP-PD-D instance from any BPP-D instance in the following steps:

1) The number of requests is the same with the number of items in the corresponding BPP-D instance, i.e., $|R| = |H|$. There is a one-to-one correspondence between requests and items. The item that corresponds to $r \in R$ is denoted as $r \in H$ hereafter.
2) Let N be the smallest integer that satisfies $\lfloor w_r N/B \rfloor \geq 1, \forall r \in R$. Bidirected graph $G = (V, L)$ is given, where $|V| = 2|R| + K + \sum_{r \in R}(\lfloor w_r N/B \rfloor - 1)$. There are three disjoint subsets of V, V_{sd}, V_{c}, and V_{m}, where $|V_{\mathrm{sd}}| = 2|R|$, $|V_{\mathrm{c}}| = K$, and $|V_{\mathrm{m}}| = \sum_{r \in R}(\lfloor w_r N/B \rfloor - 1)$. In addition, V_{m} has disjoint subset $V_{\mathrm{m}}^r = \{v_{\mathrm{m}}^{r,n} | n = 1, ..., \lfloor w_r N/B \rfloor - 1\}, \forall r \in R$. The source and destination nodes of each request are set to different nodes in V_{sd} with each other and those of other requests, i.e., $v_{\mathrm{s}}^r \neq v_{\mathrm{d}}^r, \forall r \in R$, $v_{\mathrm{s}}^r \neq v_{\mathrm{s}}^{r'}, \forall r, r' \in R, r \neq r'$, $v_{\mathrm{d}}^r \neq v_{\mathrm{d}}^{r'}, \forall r, r' \in R, r \neq r'$. There is no link between two nodes in V_{sd}, between those in V_{c}, and between a node in V_{m}^r and that in $V_{\mathrm{m}}^{r'}$, where $r, r' \in R, r \neq r'$. Nodes in V_{m}^r are concatenated in the order of index n by $(\lfloor w_r N/B \rfloor - 2)$ links. $v_{\mathrm{m}}^{r,1} \in V_{\mathrm{m}}^r$ is adjacent to the source and destination nodes of $r \in R$. $v_{\mathrm{m}}^{r,\lfloor w_r N/B \rfloor - 1} \in V_{\mathrm{m}}^r$ is adjacent to all nodes in V_{c}. Note that if $|V_{\mathrm{m}}^r| = 1$, the node in V_{m}^r is adjacent to the source and destination nodes of $r \in R$ and all nodes in V_{c}. Furthermore, if $|V_{\mathrm{m}}^r| = 0$, the source and destination nodes of $r \in R$ are adjacent to all nodes in V_{c}. Therefore, the number of hops between the source and a node in V_{c} is $\lfloor w_r N/B \rfloor$, i.e., $d(v_{\mathrm{s}}^r, v) = \lfloor w_r N/B \rfloor, \forall r \in R, v \in V_{\mathrm{c}}$. Similarly, the number of hops between the destination and a node in V_{c} is $\lfloor w_r N/B \rfloor$, i.e., $d(v_{\mathrm{d}}^r, v) = \lfloor w_r N/B \rfloor, \forall r \in R, v \in V_{\mathrm{c}}$.
An example of $G = (V, L)$ is shown in Fig. 25, where $R = \{r_1, r_2, r_3\}$, $w_{r_1} = 3$, $w_{r_2} = 2$, $w_{r_3} = 1$, $B = 3$, $K = 2$, and then $N = 3$. Since $\lfloor w_{r_1} N/B \rfloor = 3$, the path between $u \in \{v_{\mathrm{s}}^{r_1}, v_{\mathrm{d}}^{r_1}\}$ and $v \in V_{\mathrm{c}}$ has middle nodes $v_{\mathrm{m}}^1, v_{\mathrm{m}}^2 \in V_{\mathrm{m}}^{r_1}$; $d(v_{\mathrm{s}}^{r_1}, v) = d(v_{\mathrm{d}}^{r_1}, v) = \lfloor w_{r_1} N/B \rfloor = 3, v \in V_{\mathrm{c}}$. Similarly, since $\lfloor w_{r_2} N/B \rfloor = 2$, the path between $u \in \{v_{\mathrm{s}}^{r_2}, v_{\mathrm{d}}^{r_2}\}$ and $v \in V_{\mathrm{c}}$ has middle node $v_{\mathrm{m}}^3 \in V_{\mathrm{m}}^{r_2}$; $d(v_{\mathrm{s}}^{r_2}, v) = d(v_{\mathrm{d}}^{r_2}, v) = \lfloor w_{r_2} N/B \rfloor = 2, v \in V_{\mathrm{c}}$. On the other hand, since $\lfloor w_{r_3} N/B \rfloor = 1$, the path between $u \in \{v_{\mathrm{s}}^{r_3}, v_{\mathrm{d}}^{r_3}\}$ and $v \in V_{\mathrm{c}}$ has no middle node; $d(v_{\mathrm{s}}^{r_3}, v) = d(v_{\mathrm{d}}^{r_3}, v) = \lfloor w_{r_3} N/B \rfloor = 1, v \in V_{\mathrm{c}}$.
3) All requests pass through only one particular VNF, namely f, i.e., $F_r = \{f\} = F, \forall r \in R$. VNF f does not

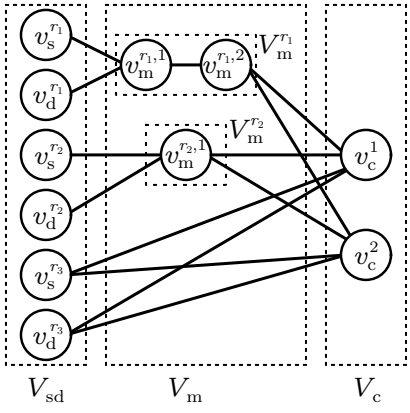


Fig. 25. Example of $G = (V, L)$ in VPP-PD-D.

change the transmission capacity.

4) The transmission capacity of each request is set to w_r, i.e., $D_r^i = w_r, \forall r \in R, i \in \{0, 1\}$.
5) The number of CPU cores that VNF f requires per unit of transmission capacity is set to $1/B$, i.e., $\Delta_f = 1/B$; the sum of the transmission capacity that is processed by VNF f cannot exceed B.
6) Nodes in $V_{\rm sd}$ and $V_{\rm m}$ have no CPU core, i.e., $c_v = 0, \forall v \in V_{\rm sd} \cup V_{\rm m}$. On the other hand, nodes in $V_{\rm c}$ have one CPU core, i.e., $c_v = 1, \forall v \in V_{\rm c}$. From this, $|V_{\rm c}| = K$, and $\Delta_f = 1/B$, there is a one-to-one correspondence between nodes in $V_{\rm c}$ in VPP-PD-D and bins in BPP-D.
7) The number of used CPU cores cannot exceed K, i.e., $I = K$.
8) The threshold of $d_{\rm sum}$ is set to $2KN$, i.e., $T = 2KN$.

The time complexity of the construction of a VPP-PD-D instance is $O(|V||R|)$.

If a BPP-D instance is a Yes instance, a given set of items can be accommodated in K bins or less; there is an item allocation such that $\sum_{h \in H_p} w_h \leq B, \forall p \in P, \bigsqcup_{p \in P} H_p = H$, where P and H_p are a set of K bins and a set of items packed in $p \in P$, respectively. Note that $H_p = \emptyset, \exists p \in P$ if a set of items is accommodated in less than K bins. From the one-to-one correspondence between requests and items and that between nodes in $V_{\rm c}$ and bins, there is a VNF placement such that $\sum_{r \in R_v} w_r \leq B, \forall v \in V_{\rm c}, \bigsqcup_{v \in V_{\rm c}} R_v = R$, where R_v is a set of requests that passes through VNF f on $v \in V_{\rm c}$ in the VPP-PD-D instance; this ensures that there is a VNF placement such that the number of used CPU cores is K or less since the transmission capacity of $r \in R$, the number of nodes where VNFs can be deployed, and the maximum accommodatable transmission capacity of $v \in V_{\rm c}$ are w_r, K, and B, respectively. From this, $\lceil \sum_{r \in R_v} (w_r/B) \rceil = 1$ if $R_v \neq \emptyset$ and 0 otherwise; $\lceil \sum_{r \in R_v} (w_r/B) \rceil$ represents whether the CPU core on $v \in V_{\rm c}$ is used or not. As the number of used CPU core cannot exceed K, $\sum_{v \in V_{\rm c}} \lceil \sum_{r \in R_v} (w_r/B) \rceil \leq K$ and then $\sum_{r \in R} (w_r/B) = \sum_{v \in V_{\rm c}} \sum_{r \in R_v} (w_r/B) \leq K$. Furthermore, $\sum_{r \in R} \lfloor w_r N/B \rfloor \leq \sum_{r \in R} (w_r N/B) \leq KN$. From the number of hops between two nodes, $d_{\rm sum} = 2 \sum_{r \in R} \lfloor w_r N/B \rfloor$. Therefore, $d_{\rm sum} \leq 2KN$; the VPP-PD-D instance is a Yes instance.

Conversely, if the constructed VPP-PD-D instance is a Yes instance, there is a VNF placement such that $d_{\rm sum}$ is $2KN$ or less. Since the number of used CPU cores cannot exceed K from the setting, this simultaneously indicates that there is a VNF placement such that the number of used CPU cores is K or less. As the node capacity constraints are satisfied, $\sum_{r \in R_v} w_r \leq B, \forall v \in V_{\rm c}, \bigsqcup_{v \in V_{\rm c}} R_v = R$. Note that $R_v = \emptyset, \exists v \in V$ if the number of used CPU cores is less than K. From the one-to-one correspondence between requests and items and that between nodes in $V_{\rm c}$ and bins, there is an item allocation such that $\sum_{h \in H_p} w_h \leq B, \forall p \in P, \bigsqcup_{p \in P} H_p = H$ in the corresponding BPP-D instance; this ensures that there is no item that cannot be packed in any bin in P, where the size of item $h \in H$, the number of bins, and the capacity of each bin are w_h, K, and B, respectively. Therefore, there is an allocation of items to K bins or less; the corresponding BPP-D instance is a Yes instance.

From the discussion above, BPP-D is reducible to VPP-PD-D; any problem in NP is reducible to VPP-PD-D since BPP-D is NP-complete. As VPP-PD-D is in NP, VPP-PD-D is NP-complete. □

Then, we analyze the hardness of RP-PD, which is formulated in the ILP problem in (40)–(42). We define the decision version of RP-PD (RP-PD-D) as follows:

Definition 3. Bidirected graph $G = (V, L)$, set of requests R, and set of VNFs F are given. Logical layered network $G^{\rm L}$ is constructed from $G = (V, L)$. The number of layers in $G^{\rm L}$ is $n_{\max} = \max_{r \in R} n_r$, where n_r is the number of required VNFs for $r \in R$. The maximum transmission capacity of $l \in L$ is b_l. The source and the destination of $r \in R$ are $v_{{\rm s}^r} \in V$ and $v_{{\rm d}^r} \in V$, respectively. The transmission capacity of $r \in R$ on the ith layer is D_r^i. The ith VNF for $r \in R$, f_i^r, is deployed on $v_{f_i^r}$. The link utilization cost per unit of transmission capacity for $l \in L$ is Ψ_l. Is there any set of request routes such that the total link utilization cost is T or less?

Theorem 3. RP-PD-D is NP-complete.

Proof. We prove the NP-completeness of RP-PD-D by using the decision version of the integer multicommodity flow problem (IMFP-D), which is NP-complete [39]. IMFP-D decides whether there is a set of flow routes such that the total cost is C or less for flow set M on graph $G = (V, L)$, where the demand of flow m is Q_m, the capacity of link l is a_l, and the utilization cost of link l is Ω_l. The total cost is $\sum_{m \in M} \sum_{l \in L} \Omega_l Q_m x_l^m$, where x_l^m is a binary variable such that $x_l^m = 1$ if $m \in M$ passes through $l \in L$ and 0 otherwise. In IMFP-D, each flow cannot be separated to multiple flows.

RD-PD-D is equivalent to IMFP-D if the settings of RD-PD-D are as follows:

1) Logical layered network $G^{\rm L}$ is constructed from graph $G = (V, L)$. $G = (V, L)$ is the same with the graph used in the IMFP-D instance.
2) The number of requests is the same with the number of flows in the corresponding IMFP-D instance, i.e., $|R| = |M|$. There is a one-to-one correspondence between requests in RP-PD-D and flows in IMFP-D. The source and destination nodes of $r \in R$ are the same

with those of the corresponding flow in M. The flow that corresponds to $r \in R$ is denoted as $r \in M$ hereafter.

3) Each request passes through only one VNF, i.e., $F_r = \{f_0^r\}, \forall r \in R$. From this, G^{L} consists of two layers, i.e., $i \in \{0, 1\}$. VNF f_0^r does not change the transmission capacity.
4) The transmission capacity of each request is set to Q_r, i.e., $D_r^i = Q_r, \forall r \in R, i \in \{0, 1\}$.
5) VNF f_0^r is deployed on the destination of $r \in R$, i.e., $v_{f_0^r} = v_{\mathrm{d}}^r, \forall r \in R$. Therefore, we only need to find a route that connects v_{s^r} and v_{d^r} on the zeroth layer of G^{L}.
6) The maximum transmission capacity of each link is a_l, i.e., $b_l = a_l, \forall l \in L$.
7) The link utilization cost is set to Ω_l, i.e., $\Psi_l = \Omega_l, \forall l \in L$.
8) The threshold of the link utilization cost is set to C, i.e., $T = C$.

Therefore, RD-PD-D is NP-complete. □

ACKNOWLEDGMENT

The authors would like to thank Prof. Ryoichi Shinkuma for discussions.

REFERENCES

[1] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 2016.

[2] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Magazine*, vol. 53, no. 2, pp. 90–97, Feb. 2015.

[3] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, Dec. 2015.

[4] W. Ma, J. Beltran, D. Pan, and N. Pissinou, "Placing traffic-changing and partially-ordered NFV middleboxes via SDN," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1303–1317, Dec. 2019.

[5] H. Huang, P. Li, S. Guo, W. Liang, and K. Wang, "Near-optimal deployment of service chains by exploiting correlations between network functions," *IEEE Trans. Cloud Comput.*, vol. 8, no. 2, pp. 585–596, Dec. 2020.

[6] N. Huin, B. Jaumard, and F. Giroire, "Optimal network service chain provisioning," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1320–1333, Jun. 2018.

[7] Z. Allybokus, N. Perrot, J. Leguay, L. Maggi, and E. Gourdin, "Virtual function placement for service chaining with partial orders and anti-affinity rules," *Networks*, vol. 71, no. 2, pp. 97–106, Mar. 2018.

[8] N. Hyodo, T. Sato, R. Shinkuma, and E. Oki, "Virtual network function placement for service chaining by relaxing visit order and non-loop constraints," *IEEE Access*, vol. 7, pp. 165 399–165 410, Aug. 2019.

[9] Y. Chen and J. Wu, "NFV middlebox placement with balanced set-up cost and bandwidth consumption," in *Proceedings of the 47th International Conference on Parallel Processing*. New York, NY, USA: Association for Computing Machinery, Aug. 2018.

[10] Y. Sumi and T. Tachibana, "Heuristic service chain construction algorithm based on VNF performances for optimal data transmission services," *IEICE Trans. Commun.*, vol. E104.B, no. 7, pp. 817–828, Jul. 2021.

[11] S. Ozaki, T. Sato, and E. Oki, "Service chain provisioning model considering traffic amount changed by virtualized network functions," in *IEEE Int. Conf. High Perform. Switching and Routing (HPSR)*, Jun. 7–10, 2021, pp. 1–6.

[12] M. Sasabe and T. Hara, "Capacitated shortest path tour problem-based integer linear programming for service chaining and function placement in NFV networks," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 104–117, Dec. 2021.

[13] Y. Xu and V. P. Kafle, "A mathematical model and dynamic programming based scheme for service function chain placement in NFV," *IEICE Trans. Inf. & Syst.*, vol. E102.D, no. 5, pp. 942–951, Jul. 2019.

[14] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 10, pp. 2179–2192, Nov. 2019.

[15] D. Li, P. Hong, K. Xue, and J. Pei, "Virtual network function placement considering resource optimization and SFC requests in cloud datacenter," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 7, pp. 1664–1677, Feb. 2018.

[16] D. Li, P. Hong, K. Xue, and J. Pei, "Availability aware VNF deployment in datacenter through shared redundancy and multi-tenancy," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1651–1664, Aug. 2019.

[17] H. Hawilo, M. Jammal, and A. Shami, "Network function virtualization-aware orchestrator for service function chaining placement in the cloud," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 643–655, Jan. 2019.

[18] M. Karimzadeh-Farshbafan, V. Shah-Mansouri, and D. Niyato, "A dynamic reliability-aware service placement for network function virtualization (NFV)," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 318–333, Dec. 2020.

[19] J. Pei, P. Hong, K. Xue, D. Li, D. S. L. Wei, and F. Wu, "Two-phase virtual network function selection and chaining algorithm based on deep learning in SDN/NFV-enabled networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1102–1117, Apr. 2020.

[20] R. Solozabal, J. Ceberio, A. Sanchoyerto, L. Zabala, B. Blanco, and F. Liberal, "Virtual network function placement optimization with deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 292–303, Dec. 2020.

[21] J. Y. Yen, "Finding the K shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, Jul. 1971.

[22] B. Y. Chen, X.-W. Chen, H.-P. Chen, and W. Lam, "Efficient algorithm for finding k shortest paths based on re-optimization technique," *Transportation Research Part E Logistics and Transportation Review*, vol. 133, p. 101819, Jan. 2020.

[23] P. Quinn, U. Elzur, and C. Pignataro, "Network Service Header (NSH)," RFC 8300, Internet Engineering Task Force, Jan. 2018. [Online]. Available: http://www.ietf.org/rfc/rfc8300.txt

[24] M. Delorme, M. Iori, and S. Martello, "Bin packing and cutting stock problems: Mathematical models and exact algorithms," *Eur. J. Oper. Res.*, vol. 255, no. 1, pp. 1–20, Nov. 2016.

[25] C. Munien and A. Ezugwu, "Metaheuristic algorithms for one-dimensional bin-packing problems: A survey of recent advances and applications," *J. Intell. Syst.*, vol. 30, no. 1, pp. 636–663, Apr. 2021.

[26] "vSRX Virtual Firewall." https://www.juniper.net/content/dam/www/assets/datasheets/us/en/security/vsrx-virtual-firewall-datasheet.pdf (accessed Aug. 24, 2021).

[27] "McAfee Virtual Network Security Platform." https://www.mcafee.com/enterprise/en-us/assets/data-sheets/ds-virtual-network-security-platform.pdf (accessed Aug. 25, 2021).

[28] L. Ruiz, R. J. Durán, I. de Miguel, N. Merayo, J. C. Aguado, P. Fernández, R. M. Lorenzo, and E. J. Abril, "Joint VNF-provisioning and virtual topology design in 5G optical metro networks," in *IEEE Int. Conf. Transparent Opt. Netw. (ICTON)*, Jul. 9–13, 2019, pp. 1–4.

[29] G. Li, H. Zhou, G. Li, and B. Feng, "Application-aware and dynamic security function chaining for mobile networks," *J. Internet Serv. Inf. Secur.*, vol. 7, pp. 21–34, Nov. 2017.

[30] D. Careglio, S. Spadaro, A. Cabellos, J. A. Lazaro, P. Barlet-Ros, J. M. Gené, J. Perelló, F. Agraz Bujan, J. Suárez-Varela, A. Pàges, J. Paillissé, P. Almasan, J. Domingo-Pascual, and J. Solé-Pareta, "Results and achievements of the alliance project: New network solutions for 5g and beyond," *Applied Sciences*, vol. 11, no. 19, Sep. 2021.

[31] "Qosmos ixEngine." https://www.qosmos.com/products/deep-packet-inspection-engine/ (accessed Jul. 23, 2022).

[32] A. Gupta, M. Farhan Habib, U. Mandal, P. Chowdhury, M. Tornatore, and B. Mukherjee, "On service-chaining strategies using virtual network functions in operator networks," *Comput. Netw.*, vol. 133, pp. 1–16, Mar. 2018.

[33] "NSFNET: A Partnership for High-Speed Networking: Final Report." https://www.merit.edu/wp-content/uploads/2019/06/NSFNET_final-1.pdf (accessed Feb. 4, 2021).

[34] "CPLEX Optimizer." https://www.ibm.com/analytics/cplex-optimizer (accessed Feb. 9, 2021).

[35] J. E. Mitchell, "Branch-and-cut algorithms for combinatorial optimization problems," *Handbook of applied optimization*, vol. 1, pp. 65–77, 2002.

[36] "Citrix NetScaler SD-WAN for WAN Optimization: It's the smart choice." https://www.citrix.com/content/dam/citrix/en_us/documents/white-paper/netscaler-sd-wan-versus-riverbed-for-wanop.pdf (accessed Jul. 12, 2021).
[37] "Announcing Video Optimizations in Akamai Image and Video Manager." https://developer.akamai.com/blog/2018/06/29/announcing-video-optimizations-akamai-image-and-video-manager (accessed Jul. 20, 2021).
[38] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.
[39] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM J. Comput.*, vol. 5, no. 4, pp. 691–703, 1976.

Shintaro Ozaki (S'21) received the B.E. in Undergraduate School of Electrical and Electronic Engineering, Kyoto University, Japan, in 2021. He is currently pursuing the master's degree in Graduate School of Informatics, Kyoto University, Japan. His research interests include network function virtualization, service chaining, resource allocation, modeling, and optimization.

Takehiro Sato (S'12-M'16) received the B.E., M.E. and Ph.D. degrees in engineering from Keio University, Japan, in 2010, 2011 and 2016, respectively. He is currently an associate professor in Graduate School of Informatics, Kyoto University, Japan. From 2011 to 2012, he was a research assistant in the Keio University Global COE Program. From 2012 to 2015, he was a research fellow of Japan Society for the Promotion of Science. From 2016 to 2017, he was a research associate in Graduate School of Science and Technology, Keio University, Japan.

Eiji Oki (M'95-SM'05-F'13) received the B.E. and M.E. degrees in instrumentation engineering and the Ph.D. degree in electrical engineering from Keio University, Yokohama, Japan, in 1991, 1993, and 1999, respectively. He was with Nippon Telegraph and Telephone Corporation (NTT) Laboratories, Tokyo, from 1993 to 2008, and The University of Electro-Communications, Tokyo, from 2008 to 2017. From 2000 to 2001, he was a Visiting Scholar at Polytechnic University, Brooklyn, New York. In 2017, he joined Kyoto University, Japan, where he is currently a Professor. His research interests include routing, switching, protocols, optimization, and traffic engineering in communication and information networks. He is a Fellow of IEEE and IEICE.