Service Chain Embedding with Maximum Flow in Software Defined Network and Application to the Next-Generation Cellular Network Architecture

Jian-Jhih Kuo*, Shan-Hsiang Shen†, Hong-Yu Kang‡, De-Nian Yang*, Ming-Jer Tsai‡ and Wen-Tsuen Chen*‡

***Abstract*—With software-defined network (SDN) and network function virtualization (NFV) techniques, we can embed the service chain consisting of a sequence of virtualized network functions (VNFs), i.e., we can determine the flow path and deploy the VNFs contained in the service chain at any place on the path. In the literature, the methods of service chain embedding bound the number of VNFs at a node, whereas the link capacities are disregarded and the amount of flows is not considered, which could cause serious congestion. In addition, according to our experiment, the process overhead on a computation node is linear to the total amount of flows processed. In this paper, we propose a method of service chain embedding to maximize the total amount of flows while bounding the process overhead of the flows on a node by its computation capability and the total amount of flows on an link by its bandwidth capacity. To our knowledge, our method is the first approximation algorithm of service chain embedding with considering flow in the literature. Simulations show our algorithm has good performance in terms of the total amount of flows.**

I. Introduction

Recently, the IP traffic from mobile terminals is growing rapidly since the mobile users tend to connect the Internet anywhere anytime via wireless networks. However, in an Long Term Evolution Advanced (LTE-A) network, which is also known as the 4G mobile network, all the traffic should pass through the packet data network gateway (P-GW) in the core network, which makes the P-GW the bottleneck of the network and limits the utilization of network bandwidth. In addition, it takes advantage of specialized network components (e.g., dedicated hardware) to perform network functions, which makes it inflexible and difficult to add new services. Therefore, the LTE-A architecture has become cumbersome to cope with the upcoming traffic surge. Therefore, the adoption of a whole new network architecture is inevitable [1]–[4].

This new architecture (termed C-SDN in [2], all-SDN in [3], and OpenNF in [5]) takes advantage of the two promising technologies in next-generation networks, software-defined network (SDN) and network function virtualization (NFV). The former one, SDN, separates the control and data planes, providing the flexibility of traffic routing while the latter one, NFV, replaces the dedicated hardware with high-volume commodity servers, leading to the elasticity of resource allocation and service deployment. With these two technologies, the allocation of the sequences of network functions and the traffic route via SDN switches becomes flexible.

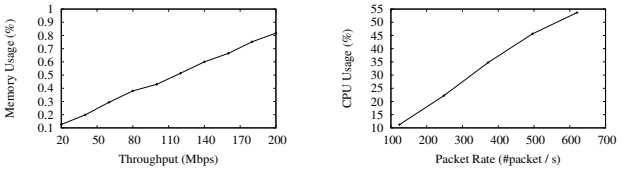
The VNF placement problem has received considerable attentions in recent years [4], [6]. However, each of these methods for the VNF placement problem does not embed the service chains, i.e., the method does not identify the routing paths of the service chains. In the literature, several methods of the service chain embedding problem have been proposed [7], [8]. These methods bound the number of VNFs at a node, whereas the link capacities are disregarded and the amount of flows is not considered, which could cause serious congestion when the total amount of flows demanded by admitted service chains is unexpectedly large. In addition, we observe that a node usually demands greater overhead to process larger amount of flow and demands different overhead to provide different types of services to process flow in actual practice.[i] Moreover, the process overhead of flows demanded by service chains on a node is accumulative.[ii] Thus, without considering flow, a method of the service chain embedding problem can only bound the number of VNFs at a node, and thus, the computation capability of a node could be oversubscribed when

*Inst. of Information Science, Academia Sinica, Taipei, Taiwan
†Dept. of Computer Science & Information Engineering, National Taiwan University of Science & Technology, Taipei, Taiwan
‡Dept. of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
E-mail:{lajacky,dnyang,chenwt}@iis.sinica.edu.tw, sshen@csie.ntust.edu.tw and {g104062629,mjtsai}@cs.nthu.edu.tw

[i]We implement two network functions, including the traffic optimizer and packet encapsulation, in Click software router, which captures packets from NIC and remove redundant bytes. The network functions run in an HP server with Intel(R) Xeon(R) CPU E3-1230 3.3GHz and 4GB memory, and the test traffics with different data rates are generated using iPerf. Based on our observation, the performance bottlenecks of the traffic optimizer and the packet encapsulation are the memory space and the CPU time, respectively. Figure 1(a) shows the result for the relation between the memory usage and the traffic data rate, where the memory usage denotes the occupied memory space divided by the total memory space. The traffic data rate varies from 20 to 200 Mbps; the memory usage changes from 0.13% to 0.82% and is linear to the traffic data rate. Figure 1(b) shows the result concerning the relation between the CPU usage and the packet arrival rate, where the CPU usage denotes the occupied CPU time divided by the total CPU time. The packet arrival rate varies from 124 to 620 packets per second; the CPU usage changes from 11% to 54% and is linear to the packet arrival rate.

[ii]We conduct an experiment to examine the relation between the combination of the number of flows processed by the traffic optimizer (≤ 2) and the number of flows processed by the packet encapsulation (≤ 2) and the total amount of flows able to be processed by the HP server. Table I shows the results. It is observed that the maximum amount of flows able to be processed by the traffic optimizer (or the packet encapsulation) is 451 (or 525) Mbps. Thus, for the flows processed by the traffic optimizer (or the packet encapsulation), the usage of the process capability of the HP server can be evaluated by the ratio of the total amount of the flows processed to 451 (or 525) Mbps. It can be seen that for any combination of the number of flows processed by the traffic optimizer and the number of flows processed by the packet encapsulation, the total usage of the process capability of the HP server is close to 100%. This implies the overhead of distributing the flows to different VNFs is negligible, and the process overhead of flows is almost accumulative.

978-1-5090-5336-0/17/$31.00 ©2017 IEEE



(a) (b)

Fig. 1. The memory and CPU usage when two network functions traffic optimizer and packet encapsulation are implemented in Click software router. (a) The relation between the memory usage of network function traffic optimizer and the traffic data rate. (b) The relation between the CPU usage of network function packet encapsulation and the packet rate.

TABLE I
EFFECT OF THE COMBINATION OF THE NUMBER OF FLOWS PROCESSED BY THE TRAFFIC OPTIMIZER (VNF1) AND THE NUMBER OF FLOWS PROCESSED BY THE PACKET ENCAPSULATION (VNF2) ON THE TOTAL AMOUNT OF FLOWS.

#flow (VNF1/VNF2)	Total Flow (VNF1)	Total Flow (VNF2)	Total Capability Usage (VNF1+VNF2)
1 / 0	451 Mbps	0 Mbps	100%
0 / 1	0 Mbps	525 Mbps	100%
2 / 0	448 Mbps	0 Mbps	99.5%
1 / 1	227 Mbps	254 Mbps	98.8%
0 / 2	0 Mbps	508 Mbps	96.9%
2 / 1	302 Mbps	170 Mbps	99.3%
1 / 2	149 Mbps	328 Mbps	95.6%
2 / 2	226 Mbps	254 Mbps	98.7%

the total amount of flows demanded by admitted service chains is unexpectedly large. These observations provide the motivation for this paper. In this paper, we embed service chains to maximize the total amount of flows of demands while ensuring that the process overhead of the flows on a node is bounded by its computation capability and the total amount of flows on a link is bounded by its bandwidth capacity. To the best of our knowledge, our method is the first approximation algorithm of the service chain embedding problem with considering flow.[iii]

The remainder of this paper is organized as follows. We first introduce the problem of service chain embedding with maximum flow (SCEMF) in Section II. Then, since the number of flow paths to be considered in the SCEMF problem is exponential, we propose a linear program with a specific form and design a separation oracle for the dual linear program to obtain a fractional solution with bounded error in Section III. In Section IV, we propose an approximation algorithm (to obtain the solution by rounding the fractional solution) with optimum approximation factor for the SCEMF problem and a heuristic algorithm that follows the approximation algorithm to improve the solution. We evaluate the performance of the algorithms by simulations in Section V and discuss the related works in Section VI. Finally, we conclude this paper in Section VII.

II. EMBEDDING OF SERVICE CHAINS WITH FLOW

Initially, we demonstrate the scenario in Section II-A. Subsequently, based on the scenario described, we formally define the problem in Section II-B. Finally, we show the hardness and optimum approximation factor of the problem in Section II-C.

A. The Scenario

Consider a software defined network comprising numerous computation nodes that are connected by bi-directional physical links, where each node and each link has limited computation capability and limited bandwidth capacity, respectively. There are several source-destination pairs with service chains. Each service chain is associated with maximum flow (demand) and the process overhead per unit of flow.[iv] Note that in SDN, the process of the flow can be cooperated and the process overhead of the flow can be shared by several computation nodes.[v] And, a service chain between a source-destination pair asks for a routing path with flow of amount not larger than the corresponding maximum flow between the source-destination pair such that the amount of flow does not exceed the remaining bandwidth capacity of each link on the routing path and the total process overhead of the flow does not exceed the total remaining computation capabilities of the nodes on the routing path, where the total process overhead of a flow is equal to the product of the amount of the flow and the process overhead per unit of the flow because the process overhead of a flow is linear to the amount of the flow.[i] Our goal is to saturate the demands (up to the corresponding maximum flow) of all source-destination pairs as many as possible.

B. The Problem

We define the problem based on the studied scenario in Definition 1.

Definition 1. Given a (undirected) network $G = (V, E)$ with link capacity $c(e) \in \mathbb{R}^+$ associated with each link $e \in E$ and node capacity $c(v) \in \mathbb{R}^+$ associated with each node $v \in V$, and given a set of source-destination pairs I with maximum flow $f(i) \in \mathbb{R}^+$ and process overhead per unit of flow $w(i) \in \mathbb{R}^+$ associated with each source-destination pair $i \in I$, the **S**ervice **C**hain **E**mbedding with **M**aximum **F**low (SCEMF) problem asks for the maximum total amount of flows between all source-destination pairs such that:

1) for each source-destination pair i, the total amount of flows between the source (termed s_i) and the destination (termed t_i), where $t_i \neq s_i$, does not exceed $f(i)$ (pair flow constraints);
2) for each node v, the total process overhead of the flows on v does not exceed $c(v)$ (node capacity constraints);
3) for each link e, the total amount of flows on e does not exceed $c(e)$ (link capacity constraints);

[iii] Remark that the service chain embedding problem with considering flow cannot be directly solved by transforming nodes to links and applying the existing algorithms for multi-commodity flow problems since for a node, the computation capability is a bound for the total process overhead of flows (evaluated by the amount of flows and the number and types of services provided) but not for the amount of flows.

[iv] A service chain consists of a sequence of services (functions) required to process the flow, and each function requires different overhead to process unit of flow. According to our experiment, the process overhead of flow is accumulative.[ii] Hence, for a service chain, the process overhead per unit of flow can be estimated by accumulating the process overhead per unit of flow of each function in the service chain. In addition, for simplicity of presentation, the process overhead refers to either the memory overhead or the CPU overhead in this paper. Our algorithm can be easily extended to the case where each service chain is associated with both the memory overhead and the CPU overhead per unit of flow.

[v] The packets of a flow can be processed by different computation nodes with the same function via a hash function on packet sequence number [9].

4) for each source-destination pair i, the number of routing paths between the source and the destination does not exceed one (path degree constraints).

Clearly, a flow only can be processed by the nodes on the routing path of the flow. Let $P(i)$ denote the set of all routing paths between source-destination pair i. Let $x_p^i = 1$ if and only if p is a routing path of the flow between source-destination pair i, and let $y_{p,v}^i$ denotes the fraction of processing $f(i)$ on node v on the routing path p. Then, the amount of flow routed on path p between source-destination pair i is $\sum_{v \in p} y_{p,v}^i \cdot f(i)$ and the process overhead of the flow between source-destination pair i on node v is $\sum_{p:p \in P(i) \wedge v \in p} y_{p,v}^i \cdot f(i) \cdot w(i)$. Thus, the total amount of flows between source-destination pair i and between all pairs is $\sum_{p \in P(i)} \sum_{v \in p} y_{p,v}^i \cdot f(i)$ and $\sum_{i \in I} \sum_{p \in P(i)} \sum_{v \in p} y_{p,v}^i \cdot f(i)$, respectively, the total amount of flows (between all pairs) on link e is $\sum_{i \in I} \sum_{p:p \in P(i) \wedge e \in p} \sum_{v \in p} y_{p,v}^i \cdot f(i)$, and the total process overhead of flows (between all source-destination pairs) on node v is $\sum_{i \in I} \sum_{p:p \in P(i) \wedge v \in p} y_{p,v}^i \cdot f(i) \cdot w(i)$ since the process overhead of flow is accumulative.[ii] The SCEMF problem is formulated as a mixed integer program (1a $-$ 1h). The object function 1a accounts for the goal to maximize the total amount of flows between all source-destination pairs. If constraints in 1e and 1g are satisfied, the number of routing path(s) for each source-destination pair i is at most one. If constraints in 1f and 1h are satisfied, a flow only can be processed by the nodes on the routing path of the flow. Constraints in 1b, 1c, and 1d ensure that the total amount of flows between source-destination pair i, the total process overhead of flows on node v, and the total amount of flows on link e are not greater than the corresponding maximum flow, node capacity, and link capacity, respectively.

$$\text{maximize} \quad \sum_{i \in I} \sum_{p \in P(i)} \sum_{v \in p} y_{p,v}^i \cdot f(i) \tag{1a}$$

$$\text{subject to} \quad \sum_{p \in P(i)} \sum_{v \in p} y_{p,v}^i \cdot f(i) \leq f(i), \ \forall i \in I \tag{1b}$$

$$\sum_{i \in I} \sum_{p:p \in P(i) \wedge v \in p} y_{p,v}^i \cdot f(i) \cdot w(i) \leq c(v), \quad \forall v \in V \tag{1c}$$

$$\sum_{i \in I} \sum_{p:p \in P(i) \wedge e \in p} \sum_{v \in p} y_{p,v}^i \cdot f(i) \leq c(e), \quad \forall e \in E \tag{1d}$$

$$\sum_{p \in P(i)} x_p^i \leq 1, \ \forall i \in I \tag{1e}$$

$$y_{p,v}^i \leq x_p^i, \ \forall i \in I, \forall p \in P(i), \forall v \in p \tag{1f}$$

$$x_p^i \in \{0,1\}, \ \forall i \in I, \forall p \in P(i) \tag{1g}$$

$$y_{p,v}^i \in [0,1], \ \forall i \in I, \forall p \in P(i), \forall v \in p \tag{1h}$$

C. The Hardness

The following results show the hardness and optimum approximation factor of the SCEMF problem. Due to the page limit, the proof of Theorem 1 is omitted and given in [10].

Theorem 1. *The SCEMF problem is NP-hard and cannot be approximated within a factor of $2^{O(\log^{1-\epsilon}|V|)}$ for any $\epsilon > 0$ unless unless NP is contained within quasi-polynomial deterministic time.*

III. The Fractional Solution with Bounded Error

Observe that in the original mixed integer program, the number of constraints and the number of variables are exponential (in the input size) due to an exponential number of paths between each source-destination pair. Thus, the linear program obtained by relaxing the constraints $x_p^i \in \{0,1\}$ to $x_p^i \in [0,1]$ cannot be solved by a linear program solver in polynomial time. Clearly, if we can obtain a relaxed linear program with a polynomial number of variables (in the input size), we can solve the linear program by designing a separation oracle for the linear program to tell whether there is a violated constraints (among an exponential number of constraints) in the linear program in polynomial time. However, such a linear program is hard to obtain because the number of paths between a source-destination pair is exponential. On the other hand, by [11], if a linear program is a standard form (see Definition 2) and has a polynomial number of constraints (except for non-negativity constraints of variables), the number of variables in its dual linear program is polynomial (see Definition 2). And, by combinatorial algorithm [12], we can obtain a near-optimum solution of the (primal) linear program in polynomial time by designing a separation oracle for the dual linear program if each coefficient on the left-hand side of inequality is not greater than the constant on the right-hand side of inequality in each constraint except for the non-negativity constraints of variables in the primal linear program (P1), and all coefficients on the left-hand-side of inequality and the constant on the right-hand-side of inequality are positive in each constraint except for the non-negativity constraints of variables in the dual linear program (P2). In Section III-A, we first obtain the primal reformulated linear program such that the primal linear program has the standard form, a polynomial number of constraints (except for non-negativity constraints of variables), and property P1 and the dual linear program has property P2. Subsequently, we design a separation oracle for the dual linear program in Section III-B.

Definition 2. [11] A standard form of a linear program is written as:

$$\begin{aligned} &\text{maximize} && c^T x \\ &\text{subject to} && Ax \leq b \\ & && x \geq 0 \text{ (non-negativity constraints of variables } x), \end{aligned}$$

where x is an $n \times 1$ variable vector, and c, b, and A denote $n \times 1$, $m \times 1$, $m \times n$ constant vectors, respectively, and its dual linear program is:

$$\begin{aligned} &\text{minimize} && b^T z \\ &\text{subject to} && A^T z \geq c \\ & && z \geq 0 \qquad \text{(non-negativity constraints of } z), \end{aligned}$$

where z is an $m \times 1$ variable vector.

A. The Reformulated Linear Program

To obtain the desired relaxed linear program, we first replace $y^i_{p,v} \in [0,1]$ (the constraints in 1h) with $y^i_{p,v} \geq 0$ (the constraints in 2f) to meet the form of non-negativity constraints of variables. The modified constraints do not change the original (mixed integer) program because $y^i_{p,v} \leq 1$ for all $i \in I$, all $p \in P(i)$, and all $v \in p$ due to the constraints in 1b, 1c, or 1d. In addition, to reduce the number of constraints (except for non-negativity constraints of variables) to be polynomial, we remove all constraints containing variables x^i_p (the constraints in 1e, 1f, and 1g) from the original program. This, however, could not bound the number of routing paths between a source-destination pair in the program. Note that $\sum_{p:p\in P(i)\wedge v\in p} y^i_{p,v} \leq \sum_{p:p\in P(i)\wedge v\in p} x^i_p \leq \sum_{p\in P(i)} x^i_p \leq 1,\ \forall i \in I,\ \forall v \in V$, where the first and third inequalities directly follows the constraints in 1f and 1e, respectively. Thus, we add constraints in 2e to the program to bound the number of routing paths between a source-destination pair. Note that since constraints in 2e directly follows constraints 1e and 1f, the obtained linear program is a relaxation of the original program. It is also noted that property P1 does not hold in the (relaxed linear) program since $f(i) \cdot w(i) \leq c(v)$ and $f(i) \leq c(e)$ do not always hold in constraints in 1c and 1d, respectively, for all $i \in I$, all $v \in V$, and all $e \in E$. Thus, we use $r(i,p,v)$ to denote the maximum flow between source-destination pair i able to be routed on path p through node v, where $p \in P(i)$ and $v \in p$; that is, $r(i,p,v) = \min\{\min_{e\in p}\{c(e)\}, f(i), \frac{c(v)}{w(i)}\}$ is the minimum among the minimum capacity of the links on the routing path p, the maximum flow between source-destination pair i, and the maximum flow between source-destination pair i able to be processed by node v for source-destination pair i. In addition, let $y^i_{p,v}$ denote the fraction of processing $r(i,p,v)$ (instead of $f(i)$ in the original program) on node v. Then, we obtain the primal (reformulated) linear program (2a $-$ 2f) as follows.

$$\text{maximize} \quad \sum_{i\in I}\sum_{p\in P(i)}\sum_{v\in p} y^i_{p,v}\cdot r(i,p,v) \tag{2a}$$

$$\text{subject to} \quad \sum_{p\in P(i)}\sum_{v\in p} y^i_{p,v}\cdot r(i,p,v) \leq f(i),\ \forall i\in I \tag{2b}$$

$$\sum_{i\in I}\sum_{p:p\in P(i)\wedge v\in p} y^i_{p,v}\cdot r(i,p,v)\cdot w(i) \leq c(v),\ \forall v\in V \tag{2c}$$

$$\sum_{i\in I}\sum_{p:p\in P(i)\wedge e\in p}\sum_{v\in p} y^i_{p,v}\cdot r(i,p,v) \leq c(e),\ \forall e\in E \tag{2d}$$

$$\sum_{p:p\in P(i)\wedge v\in p} y^i_{p,v} \leq 1,\ \forall i\in I, \forall v\in V \tag{2e}$$

$$y^i_{p,v} \geq 0,\ \forall i\in I, \forall p\in P(i), \forall v\in p \tag{2f}$$

Note that the primal linear program is a standard form and has a polynomial number of constraints (except for non-negativity constraints of variables). By Definition 2, we obtain our dual linear program (3a $-$ 3f). It is noteworthy that property P2 holds in the dual linear program.

$$\text{minimize} \quad \sum_{i\in I}\alpha_i\cdot mr(i) + \sum_{v\in V}\beta_v\cdot c(v) + \sum_{e\in E}\gamma_e\cdot c(e) + \sum_{i\in I}\sum_{v\in V}\tau_{i,v} \tag{3a}$$

$$\text{subject to} \quad \alpha_i + \beta_v\cdot w(i) + \sum_{e\in p}\gamma_e + \tau_{i,v}\cdot\frac{1}{r(i,p,v)} \geq 1,\ \forall i\in I, \forall p\in P(i), \forall v\in p \tag{3b}$$

$$\alpha_i \geq 0,\ \forall i\in I \tag{3c}$$

$$\beta_v \geq 0,\ \forall v\in V \tag{3d}$$

$$\gamma_e \geq 0,\ \forall e\in E \tag{3e}$$

$$\tau_{i,v} \geq 0,\ \forall i\in I, \forall v\in V \tag{3f}$$

B. The Separation Oracle

Given an arbitrary (fractional) solution $(\alpha,\beta,\gamma,\tau)$ to the dual linear program, the separation oracle tells whether a violated constraint exists or not. It is easy to examine whether there is a violated constraint in 3c, 3d, 3e, or 3f where the number of variables is polynomial. The challenge is tell whether there is a violated constraint in 3b. Due to the fact that both the number of source-destination pairs and the number of nodes are polynomial, it suffices to tell, for a (fixed) source-destination pair i and a (fixed) node v, whether there is a path $p \in P(i)$ through node v such that:

$$\sum_{e\in p}\gamma_e + \tau_{i,v}\cdot\frac{1}{r(i,p,v)} < 1 - \alpha_i - \beta_v\cdot w(i).$$

Then, we only need to find the most violated constraint for a (fixed) source-destination pair i and a (fixed) node v by computing:

$$\min_{p:p\in P(i)\wedge v\in p}\left\{\sum_{e\in p}\gamma_e + \tau_{i,v}\cdot\frac{1}{r(i,p,v)}\right\}. \tag{5}$$

It is not trivial to compute (5) for a source-destination pair i and a node v since the coefficient $\frac{1}{r(i,p,v)}$ is specific to path p while the number of paths p could be exponential. It is noted that the number of the values of $r(i,p,v) = \min\{\min_{e\in p}\{c(e)\}, f(i), \frac{c(v)}{w(i)}\}$ is bounded by the polynomial number of links. Let $W(r(i,p,v))$ denote the set of the values of $r(i,p,v)$. Then, for a source-destination pair i and a node v, we only need to evaluate:

$$\min_{w\in W(r(i,p,v))}\left\{\min_{p:p\in P(i)\wedge v\in p\wedge r(i,p,v)=w}\left\{\sum_{e\in p}\gamma_e + \tau_{i,v}\cdot\frac{1}{w}\right\}\right\}, \tag{6}$$

or equivalently,

$$\min_{w\in W(r(i,p,v))}\left\{\min_{p:p\in P(i)\wedge v\in p\wedge r(i,p,v)\geq w}\left\{\sum_{e\in p}\gamma_e + \tau_{i,v}\cdot\frac{1}{w}\right\}\right\}. \tag{7}$$

Note that, given a (fixed) w, $\tau_{i,v}\cdot\frac{1}{w}$ is a constant for a source-destination pair i and a node v. Thus,

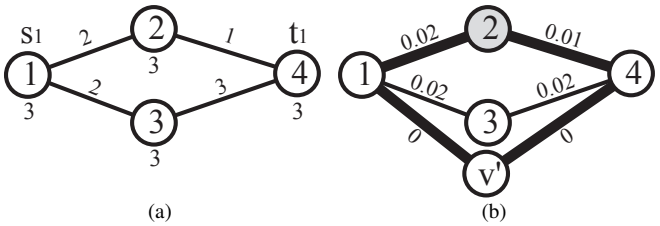


Fig. 2. Example of finding the most violated constraint in our dual linear program using Algorithm 1. (a) The input graph G. (b) The constructed graph $G_1 + v'$ with two disjoint paths p_1' and p_2', shown in bold line, obtained by Suurballe's algorithm.

$\min_{p:p\in P(i)\wedge v\in p\wedge r(i,p,v)\geq w}\left\{\sum_{e\in p}\gamma_e+\tau_{i,v}\cdot\frac{1}{w}\right\}$ can be obtained by finding a path with minimum total length (a shortest path) through v between source s_i and destination t_i in G_w, where $G_w = (V, E_w)$ denotes a (undirected) graph consisting of all nodes v and all links e with $c(e) \geq w$ in G with a length γ_e associated with each link $e \in E_w$.

Equivalently, we only need to find two disjoint paths with minimum total length between nodes v and s_i and between nodes v and t_i, respectively, in G_w. To achieve the goal, we can first obtain a graph $G_w + v'$ from G_w by adding a node v' and two links (each with an aribtrary length) between nodes v' and s_i and between nodes v' and t_i, respectively, then find two disjoint paths p_1' and p_2' with minimum total length between nodes v and v' in $G_w + v'$ by *Suurballe's algorithm* [13], and finally obtain the deisired two disjoint paths p_1 and p_2 by removing node v' and links between nodes v' and s_i and between nodes v' and t_i from paths p_1' and p_2', respectively. The separation oracle for our dual linear program is described in Algorithm 1.

Take Fig. 2 for an example, where there is only one source-destination pair $(s_1, t_1) = (1, 4)$ in the network G, $f(1) = 2.5$, $w(1) = 2$, $\alpha_1 = 0.02$, $\beta_2 = 0.02$, $\gamma_{(1,2)} = 0.02$, $\gamma_{(2,4)} = 0.01$, $\tau_{(1,2)} = 0.02$. For node 2, $W(r(1, p, 2)) = \{\min\{1, 2.5, \frac{3}{2}\}, \min\{2, 2.5, \frac{3}{2}\}, \min\{3, 2.5, \frac{3}{2}\}\} = \{1, 1.5\}$ (line 11). For $w = 1$, we construct graph G_1 (line 13), and then construct G_1+v' as shown in Fig. 2(b) (line 14). By Suurballe's algorithm, we obtain two disjoint paths $p_1' = (v', 1, 2)$ and $p_2' = (v', 4, 2)$ (line 15). Then, we obtain $p_1 = (1, 2)$ and $p_2 = (4, 2)$ (line 16), and obtain $p_1'' = (1, 2, 4)$ (line 17). Then, we have $S_{1,2,1} = \alpha_1 + \beta_2 \cdot w(1) + \sum_{e\in p_1''}\gamma_e + \tau_{1,2} \cdot \frac{1}{w} = 0.02 + 0.02 \cdot 2 + (0.02 + 0.01) + 0.02 \cdot 1 = 0.11$ (line 18) for the constraint concerning source-destination pair 1, node 2, and path p_1'' which is the most violated constraint as $S_{1,2,1} \leq \min_{v\in V, w\in W(1,p,v)}\{S_{1,v,w}\}$ (lines 19−21).

IV. The Approximation Algorithm

We first undertake the design of the LP rounding algorithm to obtain an algorithm with the optimum approximation factor (i.e., with an approximation factor of $2^{O(\log|V|)}$ by Theorem 1) for the SCEMF problem in Section IV-A. Subsequently, a heuristic algorithm is proposed to improve the solution of the SCEMF problem in Section IV-B. Finally, an example of deploying network functions in a network is demonstrated in Section IV-C. The variant of the Chernoff bound (Theorem 2) is necessary for the description of our method.

Algorithm 1 Separation Oracle for Our Dual Linear Program

input: A network $G = (V, E)$ with link capacity $c(e)$ associated with each link $e \in E$ and node capacity $c(v)$ associated with each node $v \in V$, a set of source-destination pairs I with maximum flow $f(i)$ and process overhead per unit of flow $w(i)$ associated with each source-destination pair $i \in I$, and a nonnegative (fractional) solution $(\alpha, \beta, \gamma, \tau)$

6 **begin**
7 The most violated constraint $C \leftarrow \phi$;
8 The value of the most violated constraint $S \leftarrow \infty$;
9 **forall the** *source-destination* $i \in I$ **do**
10 **forall the** *node* $v \in V$ **do**
11 $W(r(i,p,v)) \leftarrow \left\{\min\left\{c(e), f(i), \frac{c(v)}{w(i)}\right\} | e \in E\right\}$;
12 **forall the** $w \in W(r(i,p,v))$ **do**
13 Construct a graph $G_w = (V, E_w)$ with a weight (length) γ_e associated with each link $e \in E_w$, where $E_w = \{e | (e \in E) \wedge (c(e) \geq w)\}$;
14 Construct a graph $G_w + v'$ by adding a node v' and two links (with a weight 0) between nodes v' and s_i and between nodes v' and t_i, respectively, to G_w;
15 **if** *Two disjoint paths p_1' and p_2' with minimum total length between nodes v and v' can be found by Suurballe's algorithm* [13] *(i.e., p_1' and p_2' exist) in $G_w + v'$* **then**
16 Obtain two paths p_1 and p_2 by removing node v' and links between nodes v' and s_i and between nodes v' and t_i from paths p_1' and p_2', respectively; // p_1 and p_2 are two disjoints paths between nodes v and s_i and between nodes v and t_i, respecitively, in G_w
17 Obtain a path p_w'' by combining two paths p_1 and p_2; // p_w'' is a path with minimum total length through v between source s_i and destination t_i in G_w, and the total length of p_w'' is equal to $\sum_{e\in p_w''}\gamma_e$
18 $S_{i,v,w} \leftarrow \alpha_i + \beta_v \cdot w(i) + \sum_{e\in p_w''}\gamma_e + \tau_{i,v} \cdot \frac{1}{w}$;
19 **if** $S_{i,v,w} < S$ **then**
20 $C \leftarrow (i, p_w'', v)$; // the constraint concerning source-destination pair i, path p_w'', and node v is the most violated constraint
21 $S \leftarrow S_{i,v,w}$;
22 Return C;

Theorem 2. [14] *Given a set of n independent random variables, $X_1, X_2, ..., X_n \in [0, 1]$, $Pr[X > (1 + \epsilon)\mu] \leq \left(\frac{e^\epsilon}{(1+\epsilon)^{(1+\epsilon)}}\right)^\mu$ for any $\epsilon \geq 0$, where $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}(X)$.*

A. The Randomized Rounding Algorithm

Given a near-optimal (fractional) solution y' of our primal linear program, we employ a randomized rounding algorithm to obtain the solution of the SCEMF problem. First, to meet path degree constraints, at most one path can be chosen to route the flow between the source-destination i for all $i \in I$. That is, at most one variable $y_{p,v}^i$ can be rounded to one among all variables $y_{p,v}^i$ for each source-destination i. Due to the constraints in 2e, we have $\sum_{p:p\in P(i)\wedge v\in p} y_{p,v}'^i \leq 1$, $\forall i \in I, \forall v \in V$. This implies $\sum_{v\in V}\sum_{p:p\in P(i)\wedge v\in p} y_{p,v}'^i \cdot \frac{1}{|V|} \leq 1, \forall i \in I$. Therefore, if we round at most one variable $y_{p,v}^i$ to one with probability $y_{p,v}'^i \cdot \frac{1}{|V|}$

among all variables $y^i_{p,v}$[vi] and route a flow of amount $r(i,p,v)$ on path p through node v for the variable $y^i_{p,v}$ rounded to one for each source-destination pair i, the randomized solution meets path degree constraints and pair flow constraints, and the expected value of the randomized solution is greater than or equal to $\frac{1}{|V|}$ times the value of solution y'. In addition, if the amount of the flow routed on path p through node v for source-destination pair i is equal to $\frac{r(i,p,v)}{4}$, the probability of violating the capacity constraint of an arbitrary node and the probability of violating the capacity constraint of an arbitrary link each are at most $\frac{1}{|V|^4}$ as shown in Theorem 4, respectively. The proposed algorithm for the SCEMF problem is described in Algorithm 2, which is shown to be a randomized $2^{O(\log|V|)}$-approximation algorithm in Theorem 3.

Algorithm 2 Randomized $2^{O(\log|V|)}$-Approximation Algorithm for Our Problem

input: A network $G=(V,E)$ with link capacity $c(e)$ associated with each link $e \in E$ and node capacity $c(v)$ associated with each node $v \in V$, and a set of source-destination pairs I with maximum flow $f(i)$ and process overhead per unit of flow $w(i)$ associated with each source-destination pair $i \in I$

begin

Solve the linear program (2a - 2f) and obtain a (fractional) solution y' with an error bound ω using combinatorial algorithm [12] which employs Algorithm 1 as the separation oracle for our dual linear program;

For each source-destination pair i, round at most one variable $y^i_{p,v}$ to one with probability $\frac{y'^i_{p,v}}{|V|}$ among all variables $y^i_{p,v}$;

For each source-destination pair i, route a flow of amount $\frac{r(i,p,v)}{4}$ on path p through node v for the variable $y^i_{p,v}$ rounded to one;

Return the solution;

Theorem 3. *The proposed algorithm is a randomized $2^{O(\log|V|)}$-approximation algorithm for the SCEMF problem.*

Proof: Let OPT and OPT_{PL} be the optimum values of our mixed integer program (the SCEMF problem) and our primal linear program, respectively. Since constraints in 2e directly follows constraints 1e and 1f, our primal linear program is a relaxation of our mixed integer program. Thus, $OPT \le OPT_{PL}$. In addition, let y" be the solution output by Algorithm 1, and let $value(y')$ and $value(y")$ be the values of y' (obtained in step 2) and y", respectively. Due to the constraints in 2e, we have $\sum_{p:p\in P(i)\wedge v\in p} y'^i_{p,v} \le 1$, $\forall i \in I, \forall v \in V$. This implies $\sum_{v\in V}\sum_{p:p\in P(i)\wedge v\in p} y'^i_{p,v}\cdot\frac{1}{|V|} \le 1$, $\forall i \in I$. Since we round at most one variable $y^i_{p,v}$ to one with probability $y'^i_{p,v}\cdot\frac{1}{|V|}$ among all variables $y^i_{p,v}$ and route a flow of $\frac{r(i,p,v)}{4}$ on path p through node v for the variable $y^i_{p,v}$ rounded to one for each source-destination pair i, $\mathbb{E}[value(y")] \ge \frac{value(y')}{4|V|}$. Since $value(y') \ge \frac{OPT_{PL}}{1+\omega}$ by [12], $\mathbb{E}[value(y")] \ge \frac{OPT}{4|V|\cdot(1+\omega)} = \frac{OPT}{O(|V|)} = \frac{OPT}{2^{O(\log|V|)}}$. ■

By Algorithm 2, at most one flow of amount $\frac{r(i,p,v)}{4} < f(i)$ is routed between each source-destination pair i. Thus, both the pair flow constraints and the path degree constraints are satisfied. Concerning the node (or link) capacity constraints, let z^i_v (or z^i_e) be a random variable denoting the total process overhead of flows on node v (or the total amount of flows on link e) for source-destination pair i. Then,

$$z^i_v = \begin{cases} \frac{r(i,p,v)}{4}\cdot w(i) & \text{with probability } \frac{y'^i_{p,v}}{|V|}, \\ & \forall p \in \{p | p \in P(i) \wedge v \in p\}; \\ 0 & \text{otherwise;} \end{cases}$$

and,

$$z^i_e = \begin{cases} \frac{r(i,p,v)}{4} & \text{with probability } \frac{y'^i_{p,v}}{|V|}, \\ & \forall v \in V, \forall p \in \{p | p \in P(i) \wedge v \in p \wedge e \in p\}; \\ 0 & \text{otherwise.} \end{cases}$$

Let $z_v = \sum_{i\in I} z^i_v$ be a random variable denoting the total process overhead of flows on node v, and let $z_e = \sum_{i\in I} z^i_e$ be a random variable denoting the total amount of flows on node e. Then, the expectation of the total process overhead of flows on node v, $\mathbb{E}[z_v]$, is at most $\frac{c(v)}{4|V|}$ and the expectation of the total amount of flows on link e, $\mathbb{E}[z_e]$, is at most $\frac{c(e)}{4|V|}$, as described in Lemma 1. In addition, the probability of violating the capacity constraint of node v, $Pr[z_v > c(v)]$, and the probability of violating the capacity constraint of link e, $Pr[z_e > c(e)]$, each are at most $\frac{1}{|V|^4}$, as described in Theorem 4. Since the number of nodes (or links) in G are at most $|V|$ (or$|V|^2$), the probability of violating at least one node (or link) capacity constraint is less than $\frac{1}{|V|^3}$ ($\frac{1}{|V|^2}$), which is negligible.

Lemma 1. $\mathbb{E}[z_v] \le \frac{c(v)}{4|V|}$ *and* $\mathbb{E}[z_e] \le \frac{c(e)}{4|V|}$.

Proof: Clearly,

$$\mathbb{E}[z_v] = \sum_{i\in I}\sum_{p:p\in P(i)\wedge v\in p} \frac{y'^i_{p,v}}{|V|}\cdot\frac{r(i,p,v)}{4}\cdot w(i) \le \frac{c(v)}{4|V|},$$

where the last inequality directly follows the node capacity constraints in 2c, and

$$\mathbb{E}[z_e] = \sum_{i\in I}\sum_{p:p\in P(i)\wedge e\in p}\sum_{v\in p} \frac{y'^i_{p,v}}{|V|}\cdot\frac{r(i,p,v)}{4} \le \frac{c(e)}{4|V|},$$

where the last inequality directly follows the link capacity constraints in 2d. ■

Theorem 4. $Pr[z_v > c(v)] \le \frac{1}{|V|^4}$ *and* $Pr[z_e > c(e)] \le \frac{1}{|V|^4}$.

Proof: The proof of $Pr[z_e > c(e)] \le \frac{1}{|V|^4}$ is omitted due to its similarity with that of $Pr[z_v > c(v)] \le \frac{1}{|V|^4}$.

[vi] We treat that variable $y^i_{p,v}$ is rounded to one as an event with probability $y'^i_{p,v}\cdot\frac{1}{|V|}$ for each variable $y^i_{p,v}$, and that no variable is rounded to one as an event with probability $1 - \sum_{v\in V}\sum_{p\in\{p|P(i)\wedge v\in p\}} \frac{y'^i_{p,v}}{|V|}$. We decide which event occurs based on the outcome of a trail [15]. For example, at most one of two variables y_1 and y_2 are to be rounded to one with probabilities 0.5 and 0.2, respectively. We first generate a uniformly-random value x in the range $[0,1)$. Then, y_1 is rounded to one if $x \in [0,0.5]$; y_2 is rounded to one if $x \in (0.5,0.7]$; otherwise, none of y_1 and y_2 is rounded to one.

Let $z'^i_v = 4z^i_v$ be a random variable. It suffices to show $Pr[\sum_{i\in I} \frac{z'^i_v}{c(v)} > 4] \leq \frac{1}{|V|^4}$. Note that random variables $\frac{z'^i_v}{c(v)}$ for all $v \in V$ are independent; $\frac{z'^i_v}{c(v)} \in [0,1]$ because $c(v) \geq r(i,p,v)$; $\mathbb{E}[\sum_{i\in I} \frac{z'^i_v}{c(v)}] \leq \frac{1}{|V|}$ by Lemma 1. Thus, by Chernoff bound (Theorem 2) and setting $\epsilon = 4|V|-1$, we have

$$Pr[\sum_{v\in V} \frac{z'^i_v}{c(v)} > (4|V|)(\frac{1}{|V|})] \leq \left(\frac{e^{(4|V|-1)}}{(4|V|)^{(4|V|)}}\right)^{\frac{1}{|V|}} = \frac{e^4}{e^{\left(\frac{1}{|V|}\right)}(4|V|)^4} < \frac{e^4}{(4|V|)^4} < \frac{1}{|V|^4}.$$

■

Clearly, it takes $O(|I|)$ time to select paths and route flows on the selected paths for all source-destination pairs at steps 3 and 4, respectively. At step 2, combinatorial algorithm [12] iteratively employs our separation oracle (Algorithm 1) to identify the most violated constraint in our dual linear program in order to adjust the primal solution and dual solution. According to [12], at most $O(\omega^{-2}m_1 \log m_1)$ iterations are executed, where ω denote the error bound of the fraction solution and m_1 denotes the number of constraints (except for the non-negativity constraints of variables) in the primal linear programming, i.e., $m_1 = O(|I| + |V| + |E| + |I| \cdot |V|)$. As for Algorithm 1, a graph G_w is generated for each source-destination pair, each node, and each possible link weight, and thus, at most $m_2 = O(|I|\cdot|V|\cdot|E|)$ graphs are generated. For each graph G_w, we use Suurablle's algorithm to find the disjoint paths with time complexity $m_3 = O(|E| + |V|\log|V|)$ [13]. To sum up, the time complexity of Algorithm 2 is $O(\omega^{-2}m_1m_2m_3 \log m_1)$.

B. Heuristic Algorithm for Solution Improvement

Since Algorithm 2 may route zero flow for a source-destination pair in the randomized rounding process (line 3) even if there exists a routing path on which all links and nodes are not saturated between the source-destination pair, we propose a heuristic algorithm that follows Algorithm 2 to improve the solution. We first establish a path set P' by selecting the path p chosen to route the flow between the source-destination pair i by Algorithm 2 if p exists, and the path p with maximum $\sum_{v\in p} y'^i_{p,v} \cdot r(i,p,v)$ among all paths between the source-destination pair i otherwise. Subsequently, we compute the optimum solution of the reformated linear program (2a − 2f) with $P(i)$ replaced with $P(i) \cap P'$. Such a linear programming can be solved in polynomial time since there are a polynomial number of constrains and variables. Note that Algorithm 2 followed by the heuristic algorithm outputs an approximation solution to the SCEMF problem.

C. Deployment of Virtual Network Functions

We demonstrate how to deploy the network functions in a network using our solution. Consider an example of a service chain that defines the flow to be first steered to the firewall, then billing, and finally destination, where the process overhead per unit of flow is assumed to be 0.1 and 0.01 (CPU usage/Mbps) for the firewall and billing network functions, respectively. Assume that our method decides a 10-Mbps flow routed on the path $s \to a \to b \to c \to t$ for the service chain, where the fraction of the total process overhead of the flow is 0.5, 0.55, and 0.05 on nodes a, b, and c, respectively. Then, based on a hash function on packet sequence number,[v] we can deploy the firewall network function on nodes a and b and the billing network function on nodes b and c, and process the first half of the flow using the firewall network function on node a, the second half of the flow using the firewall and billing network functions on node b, and the first half of the flow using the billing network function on node c. The result is illustrated in Figure 3.

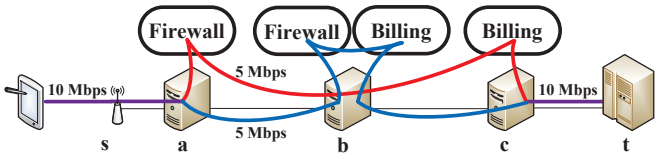


Fig. 3. Example of the deployment of the firewall and billing network functions and the amount of flow processed by the network function on a specific node.

V. Numerical Result

We conduct simulations on more than 200 realistic network topologies collected from around the world by [16]. The nodes and links in the networks ranges from 3 to 196 and from 4 to 245, respectively. We assume that the performance bottleneck of the network is the memory space (MB), and the capacities of the nodes and links in the network is 1024 MB and 100 Mbps, respectively, in the base case. In each network, we select $|M|$ source-destinations pairs at random, and generate the flows for the selected source-destination pairs based on a power law distribution [17] with the probability density function, $P[X = x]$ is equal to $Cx^{-\alpha}$ if $x \geq x_{min}$, 0 otherwise, where x_{min} denotes the minimum amount of fow, and $C = (\alpha - 1)x_{min}^{(\alpha-1)}$. In the base case, we set $x_{min} = 1$ Mbps, $\alpha = 2.1$, and $|M|$ to the number of links in the network. According to our experiment, the traffic optimizer (the network function implemented in Click software router) demands 15 MB memory to process the 1 Mbps flow. Since a flow is processed by one or multiple functions, we assume that each source-destination pair is associated with the process overhead per unit of flow randomly chosen from the range $[1, 100]$ (MB/Mbps). Based on the base case, the different settings of the node capacity, the link capacity, the number of source-destination pairs (number of flows), and the value of x_{min} are used to examine their impacts on the performance of our algorithm. We compare the solution obtained by our algorithm (i.e., Algorithm 2 followed by the heuristic algorithm for solution improvement) with the optimal solution obtained by a branch-and-bound algorithm or the optimal solution of the relaxed linear programming (2a − 2f) (whose object value is an upper bound of the optimal value for our problem) in terms of the total throughput. Since our problem is NP-hard, the branch-and-bound algorithm is only used to obtain the optimal solution for the small-scale networks which are collected into

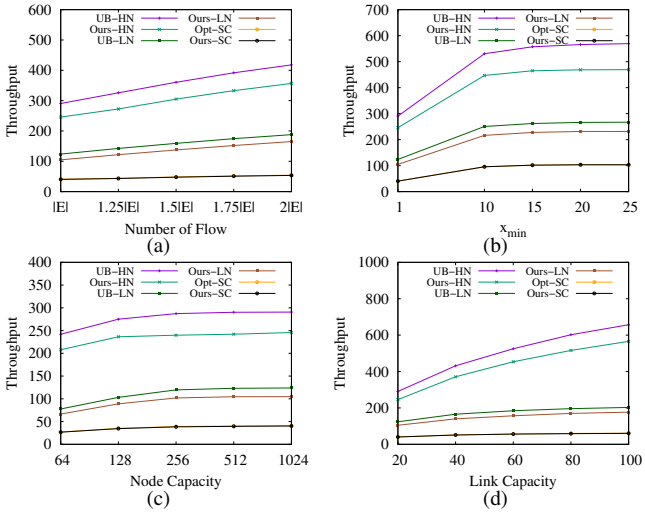


Fig. 4. Effects of the (a) number of flows, (b) x_{min}, (c) node capacity, and (d) link capacity on the throughput in the network groups HN, LN, and SC.

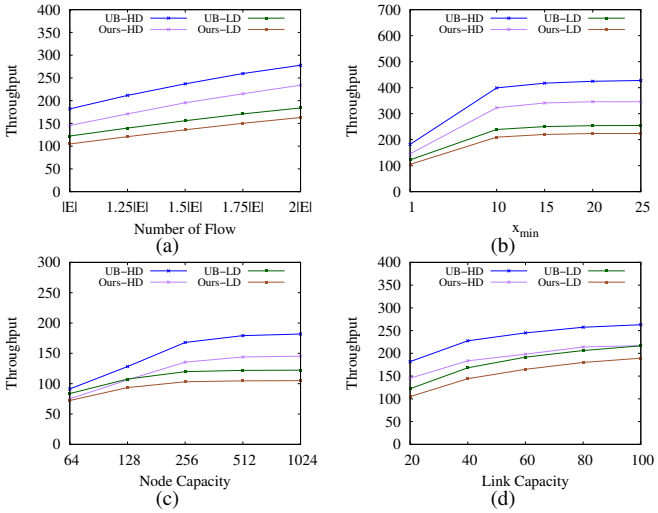


Fig. 5. Effects of the (a) number of flows, (b) x_{min}, (c) node capacity, and (d) link capacity on the throughput in the network groups HD and LD.

a group denoted by SC. We also classify all networks into two groups in two different ways. The first (or second) one classifies the networks into one group of the networks with at least 100 nodes (or average node degree at least 2.4), denoted HN (or HD), and one group of the networks with less than 100 nodes (or average node degree less than 2.4), denoted LN (or LD), where the average node degree denotes the average number of neighbors per node and the higher average node degree indidates the more diverse routing paths. In addition, we also evaluate the performance of our algorithm in terms of the node capacity usage.

Figures 4−6 show the simulations results, where our algorithm, the optimal value, and the upper bound of the optimal value are denoted by Ours, Opt, and UB, respectively. As anticipated, our algorithm (as well as UB) has greater throughput in the network group HN than in the network group LN. On the other hand, as compared to in the network groups LN, HD, LD, and SC, our algorithm has smaller node usage in the network group HN since the process overhead is shared

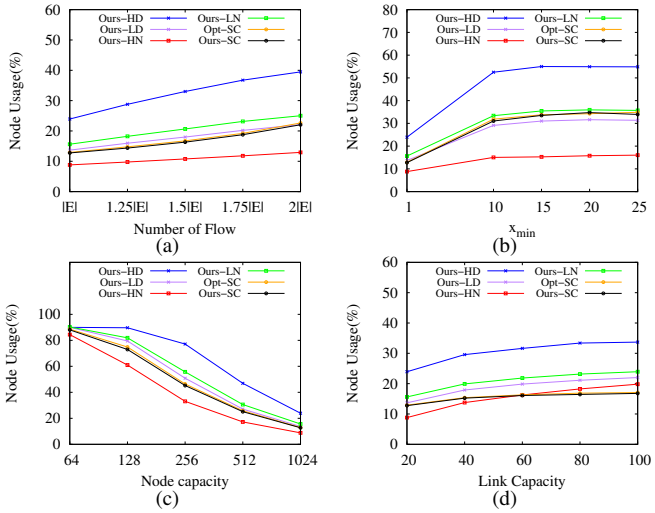


Fig. 6. Effects of the (a) number of flows, (b) x_{min}, (c) node capacity, and (d) link capacity on the node usage.

by more nodes. In addition, as compared to in the network group LD, our algorithm has greater throughput and node usage in the network group HD due to more diverse routing paths used to route the flow. Moreover, as anticipated, the throughput increases as the number of flows, x_{min}, node capacity, or link capacity increases, and the node usage increases as the number of flows, x_{min}, or link capacity increases. It is observed that as the node capacity increases, the node usage decreases. This observation results from that the process overhead increases more slowly than does the node capacity.

VI. Related Works

In [4], [6], the VNF placement problem has been studied. In [6], the authors propose efficient approximation algorithms to minimize the total cost of deploying VNFs while ensuring the number of VNFs at a node does not exceed the limitation. In [4], the authors investigate the placement problem of two types of VNFs, virtualized PGW (vPGW) and virtualized SGW (vSGW), such that the latency between vPGW and its responsible vSGW is bounded. Since these methods only address the VNF deployment but not determine the routing paths of the service chains, they cannot be used to solve our problem. In [7], [8], the service chain embedding problem that considers both of the VNF placement and routing path selection is studied. In [7], an admitted service chain must have a selected length-bounded (hop-count-bounded) routing path and the required VNFs deployed at the nodes on the path, where each node has limited capacity to host the VNFs. And, the authors present an efficient on-line algorithm to maximize the number of admitted service chains while ensuring the number of VNFs at a node does not exceed the limitation. In [8], the authors consider the networks consisting of optical and electrical domains. Since the conversion between the optical domain and the electrical domain is expensive, the authors place the VNFs of a service chain at the nodes in the same domain. Hence, the goal is to minimize the total number of domains crossed by the service chains while ensuring the limited node capacity is not violated. Remark that these algorithms for the service chain embedding

problem does not consider the flow and cannot be used to obtain an approximation algorithm for our problem. In [18], [19], the service chain embedding problem with considering the fow is studied. The proposed methods directly take advantage of CPLEX to solve the modeled mathematical programming instead of presenting an efficient algorithm (i.e., in polynomial time), which makes it hard to be used when the topology size is large. In [20], the authors aim at striking a balance between link utilization and server usage to maximize the total benefit. They deploy the service chains based on the guideline, determined based on the network conditions and demand properties, of the proper routing path length and resource reuse factor. In [21], the authors present an auction mechanism which efficiently makes the users bid the service chain truthfully and approximate the social welfare maximization for service chains in the NFV market. Since the goals of these papers are different from that of ours, these methods cannot be used to solve our problem. In [12], [22], the maximum multi-commodity flow problem, which is to maximize the total amount of flows between all source-destination pairs, has been approximated within a factor of $(1 + \omega)$, where $\omega > 0$. In [23], [24], the maximum multi-commodity flow problem with considering the node capacity has been studied, where the capacity of a node bounds the number of flow paths through the node, which is different from that of our problem. Thus, these methods cannot be used to obtain the approximation algorithm for our problem.

VII. Conclusion

In this paper, we investigated the service chain embedding problem in depth. To model the problem precisely, we conduct the experiment to examine the relation between the process overhead and the amount of flows processed on a computation node. According to our experiment, the process overhead on a computation node is accumulative and linear to the total amount of flows processed; thus, based on the experimental results, we presented the service chain embedding with maximum flow problem (SCEMF) to maximize the total amount of flows of demands while ensuring that the total process overhead of the flows on a node is bounded by its computation capability and the total amount of flows on a link is bounded by its bandwidth capacity. We showed the hardness of the SCEMF problem and proposed an approximation algorithm with the best theoretical approximation ratio. To the best of our knowledge, our method is the first approximation algorithm of the service chain embedding problem with considering flow.

To evaluate the performance of our algorithm, we conducted simulations on more than 200 realistic network topologies collected from around the world, where the synthetic maximum flows associated with source-destination pairs were generated based on the power law distribution, and the different settings of the node capacity, the link capacity, the number of flows, and the minimum amount of flow are used to examine their impacts on the performance of our algorithm. The simulation results showed the object value (the total amount of flows) of our solution is close to an upper bound of the optimal value (or an optimum value for the small-scale networks).

Acknowledgment

This research is supported by the Thematic Research Grant, Academia Sinica, Taiwan.

References

[1] A. Yegin, J. Park, K. Kweon, and J. Lee, "Terminal-centric distribution and orchestration of ip mobility for 5G networks," *Communications Magazine, IEEE*, vol. 52, pp. 86–92, 2014.
[2] A. Bradai, K. Singh, T. Ahmed, and T. Rasheed, "Cellular software defined networking: a framework," *Communications Magazine, IEEE*, vol. 53, pp. 36–43, 2015.
[3] V. Yazc, U. Kozat, and M. Oguz Sunay, "A new control plane for 5G network architecture with a case study on unified handoff, mobility, and routing management," *Communications Magazine, IEEE*, vol. 52, pp. 76–85, 2014.
[4] A. Basta, W. Kellerer, M. Hoffmann, H. J. Morper, and K. Hoffmann, "Applying nfv and sdn to lte mobile core gateways, the functions placement problem," in *ACM SIGCOMM Workshop All Things Cellular*, 2014.
[5] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "OpenNF: Enabling innovation in network function control," in *ACM SIGCOMM*, 2014.
[6] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *IEEE INFOCOM*, 2015.
[7] T. Lukovszki and S. Schmid, "Online admission control and embedding of service chains," in *SIROCCO*, 2015.
[8] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for nfv chaining in packet/optical datacenters," *Lightwave Technology, Journal of*, vol. 33, pp. 1565–1570, 2015.
[9] Service chain load balancing with opencontrail. [Online]. Available: http://www.opencontrail.org/service-chain-load-balancing-with-opencontrail/
[10] Tech. Rep. [Online]. Available: http://dclab.cs.nthu.edu.tw/~mjtsai/files/paper/INFOCOM_2017_TR.pdf
[11] T. H. Cormen, *Introduction to algorithms*. MIT press, 2009.
[12] N. Garg and J. Könemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," *SIAM J. Comput.*, vol. 37, pp. 630–652, 2007.
[13] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, 1984.
[14] D. P. Williamson and D. B. Shmoys, *The Design of Approximation Algorithms*. Cambridge University Press, 2011.
[15] Darts, dice, and coins: Sampling from a discrete distribution. [Online]. Available: http://www.keithschwarz.com/darts-dice-coins/
[16] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *Selected Areas in Communications, IEEE Journal on*, vol. 29, pp. 1765 –1775, 2011.
[17] X. Li and C. Qian, "Low-complexity multi-resource packet scheduling for network functions virtualization," in *IEEE INFOCOM*, 2015.
[18] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *CNSM*, 2014.
[19] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," 2015. [Online]. Available: https://hal.inria.fr/hal-01170042/file/optimal-routing-virtual.pdf
[20] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," in *IEEE INFOCOM*, 2016.
[21] S. Gu, Z. Li, C. Wu, , and C. Huang, "An efficient auction mechanism for service chains in the NFV market," in *IEEE INFOCOM*, 2016.
[22] L. Fleischer, "Approximating fractional multicommodity flow independent of the number of commodities," in *IEEE FOCS*, 1999.
[23] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "On the effect of forwarding table size on sdn network utilization," in *IEEE INFOCOM*, 2014.
[24] X.-N. Nguyen, D. Saucez, C. Barakat, and T. Turletti, "OFFICER: A general optimization framework for openflow rule allocation and endpoint policy enforcement," in *IEEE INFOCOM*, 2015.