This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

An Online Algorithm for VNF Service Chain Scaling in Datacenters

Ziyue Luo, *Student Member, IEEE*, and Chuan Wu, *Senior Member, IEEE, Member, ACM*

***Abstract*— Built on top of virtualization technologies, network function virtualization (NFV) provides flexible and scalable software implementation of various network functions. Virtual network functions (VNFs), which are network functions implemented as virtual machines, are chained together to provide network services. Dynamic deployment of VNFs while satisfying incoming network traffic demand is the key to cost optimization of an NFV system. Besides considering server resource capacity and incoming traffic rates, an optimal scaling policy needs to strike a balance between VNF's operational costs, the costs for maintaining VNF instances, and VNF deployment costs, additional costs when setting up new VNF instances on a server. This paper targets dynamic scaling of VNF instances in a cloud data center where multiple VNF chains are running. We propose an online scaling algorithm to adjust the deployment of VNF instances according to time-varying traffic demand, ensuring a good competitive ratio. Through theoretical analysis and trace-driven simulation, we demonstrate effectiveness of the proposed online VNF scaling algorithm.**

***Index Terms*— Network function virtualization, service chains, online algorithm, dynamic scaling.**

I. Introduction

NETWORK function virtualization is a new paradigm that aims at more flexible and scalable provisioning of softwarized network functions. While traditional network functions are implemented by expensive dedicated hardware, NFV leverages virtualization technologies to deploy virtual network functions onto commodity hardware [1]. Multiple VNFs are commonly chained (*i.e.*, VNF service chains) together to provide a network service, *e.g.*, "Web Application Firewall → IDS → Load Balancer" for access control in Web service.

A typical NFV proposal is to implement VNFs as virtual machines (VMs) or containers in a cloud computing platform, or the provider's cloud cluster. The goal of NFV is to provide network service with significant cost reduction, as compared to using traditional hardware-based middleboxes. The significant flexibility introduced by NFV with softwarized VNFs enables the network operator to initialize, terminate and migrate VNFs on the go according to demand, for substantial cost reduction. Ideally, more VNF instances should be set up in response to a rise in network traffic; in the case of decreasing network traffic rates, idle VNF instances can be terminated.

This paper studies the problem of online VNF scaling within a cloud data center. In contrast to the offline setting where traffic rates in all time slots are known, the key challenge in such online scaling lies in the unknown network traffic rates in future time slots. Further, when a network flow traverses a VNF in a service chain, the traffic rate may change, *e.g.*, deduplication or compression function is deployed in WAN optimization which can reduce traffic rates of flows bypassing it. The online algorithm should always deploy sufficient VNF instances while addressing network traffic fluctuation as well as cost minimization; it should avoid terminating VNF instances while immediately creating new ones (due to traffic rising again), even without known future flow rates.

We strategically design an online VNF scaling algorithm, tending to various practical considerations of VNF deployment in a datacenter, including server resource constraints, VNF deployment and operating costs. Our main contribution in this algorithm design is summarized as follows:

▷ We thoroughly investigate the problem of online VNF scaling in a cloud data center and formulate the problem as an integer linear program (ILP). The ILP enables dynamic VNF scaling and deployment across servers in consideration of incoming flow rate fluctuation, rate change after VNF processing, traffic routing among multiple servers, VNF deployment/operating costs and server capacities. We further transform the ILP into an equivalent simplified ILP, which serves as the foundation for online algorithm design.

▷ We leverage the regularization technique adapted from online learning literature [2] to decouple the original online ILP into a series of regularized sub-problems. We identify that the deployment cost constraint serves as the connection between consecutive time slots and lift it into the objective function for decoupling. We relax each offline sub-problem as a linear program (LP) and solve it efficiently using a convex optimization algorithm to obtain a fractional solution.

▷ We adopt a novel online rounding algorithm based on linear algebra and dependent rounding [3]. Our rounding algorithm rounds a fractional solution to an integer solution while doing its best to satisfy both the packing and covering constraints in the original problem. Our rounding algorithm together with online regularization-based algorithm guarantees a good expected competitive ratio as compared with the offline optimum of the original problem. Our rounding algorithm ensures enough VNF instances for network traffic processing while the resource capacity constraints may be violated by a

Manuscript received June 18, 2019; revised January 12, 2020; accepted February 11, 2020; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor T. He. This work was supported in part by the Research Grants Council (RGC) of Hong Kong under Contract HKU 17204619, Contract 17225516, Contract C7036-15G(CRF), and Contract C5026-18G (CRF). *(Corresponding author: Chuan Wu.)*

The authors are with the Department of Computer Science, The University of Hong Kong, Hong Kong (e-mail: zyluo@cs.hku.hk; cwu@cs.hku.hk).

Digital Object Identifier 10.1109/TNET.2020.2979263

1063-6692 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Wollongong. Downloaded on April 04,2020 at 13:46:08 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

relatively small constant. We discuss the implication of this violation in practice through evaluation.

The rest of the paper is organized as follows. We discuss related work in Sec. II and introduce the system model in Sec. III. Sec. IV presents our regularization-based online algorithm and the rounding algorithm is given in Sec. V. Sec. VI discusses the complete online algorithm. Sec. VII gives evaluation results, and Sec. VIII concludes the paper.

II. Related Work

A. Enabling Technologies for NFV

There are recent works focusing on building practical software-based networks with promising performance, *e.g.*, CoMb [4], ClickOS [5] and NetVM [6]. For management of VNF instances, Qazi *et al.* [7] implement an SDN-based policy enforcement layer, SIMPLE, aiming at network traffic steering within middleboxes. Claymen *et al.* [8] design an orchestrator for network virtual node placement and network service scheduling. A distributed orchestrater is implemented for VNF creation and removal. FreeFlow, an NFV system based on Split/Merge [9], enables transparent, balanced elasticity for virtual middleboxes that can be scaled dynamically. OpenNF [10] is a control panel architecture that achieves loss-free and order-preserving flow state migration through observing VNF state changes and a two-phase scheme for network forwarding state update. E2 [11] is a coherent framework for NFV packet processing that performs scalable, application-agnostic scheduling. The above studies pave the way for design and implementation of VNF scaling algorithms.

B. Optimal Placement of VNFs

There have been some efforts on designing efficient algorithms for optimal VNF placement. VNF-P [12] presents a resource allocation model for NFV systems in a hybrid network system containing dedicated hardware and VNFs. Ghaznavi *et al.* [13] propose a local search heuristic for geo-distributed VNF chain placement aiming at minimal consumption of physical resources. Cohen *et al.* [14] design an approximation algorithm to minimize the distance cost between clients and corresponding VNFs with theoretically proven performance. Ma *et al.* [15] address the placement challenge for NFV middleboxes with different dependency relations, *e.g.*, totally-ordered, partially-ordered, or non-ordered middlebox set. Two heuristics are proposed considering scenarios where the flow path is predetermined or not. However, these studies only perform one-time VNF placement without considering network flow fluctuation.

Shi *et al.* [16] leverage MDP to model dynamic VNF resource allocation and propose a heuristic algorithm based on resource reliability prediction by Bayesian learning. Wang *et al.* [17] design online algorithms for single VNF service chain scaling and multiple VNF service chains scaling, respectively. Jia *et al.* [18] develop an online VNF scaling algorithm leveraging the regularization method and dependent rounding technique. The algorithm is analyzed theoretically with a worst-case performance guarantee. Fei *et al.* [19] study proactive VNF provisioning considering a single type of resource. The authors leverage an online learning method to predict upcoming traffic and design an adaptive scaling strategy for resource saving and cost minimization.

Our work serves as a complementary study to previous work. We design a general online VNF scaling algorithm that practically considers multiple resource constraints, network traffic fluctuation and multiple VNF service chains.

III. System Model

A. NFV System

Consider an NFV provider who rents computational resources in a cloud data center to deploy S VNF service chains. There are in total N types of VNFs available for constructing service chains. For each service chain s, the VNF connectivity is denoted by $e^s_{n,n'}$, with $e^s_{n,n'} = 1$ if VNF n is the predecessor of VNF n' in service chain s and $e^s_{n,n'} = 0$, otherwise, $\forall n \in [N], n' \in [N]$.[1]

The system works in a time-slotted fashion, over a potentially large span of T time slots. The duration of a time slot typically ranges from minutes to hours. Input flow rates to service chain s vary from time to time, denoted by $f^s(t), t \in [T]$. In addition, we consider that the flow rate may change when traversing each VNF. We use $\lambda^s_n(t)$ to represent the flow rate change ratio of VNF n in service chain s at t. Let $\bar{\lambda}^s_n(t)$ denote the cumulative flow change ratio from the first VNF to the immediate predecessor of VNF n, VNF n^-, in service chain s: $\bar{\lambda}^s_n = \lambda^s_{n_1}(t)\lambda^s_{n_2}(t)\dots\lambda^s_{n^-}(t)$.

There are M servers and 2 types of key resources, *e.g.* CPU and memory in the cloud datacenter. Without loss of generality, we assume that the servers are homogeneous with a resource capacity of C_k for resource $k \in \{0, 1\}$. Let P_n denote the processing capability of one instance of VNF $n \in [N]$, in terms of the incoming flow rate that it can handle in a time slot. We assume different flows belonging to different service chains may share the same instance of the same type VNF n, if they place VNF n in the same server, to minimize the number of instances deployed. Each instance of VNF n requires $c_{n,k}$ amount of resource k.

At the beginning of each time slot t, the NFV provider adjusts VNF deployment for all service chains. We use $x_{m,n}(t)$ to denote VNF placement decision: $x_{m,n}(t)$ indicates the number of VNF n deployed on server m at t. We use $y^s_{m,n,m',n'}(t)$ as the routing variable: $y^s_{m,n,m',n'}(t)$ represents the amount of traffic in service chain s forwarded from VNF n in server m to VNF n' in server m' during time slot t.

B. Cost Structure

The goal of the NFV provider is to minimize the overall cost of running the service chain to serve the flows. We consider two types of costs.

1) Operating Costs: Let O_n be the operating cost for running each instance of VNF n per time slot, e.g., for renting a virtual machine or container with required resource configuration for running the instance. The overall operational cost for running all VNFs of the service chains at t is:

$$C_{operate}(t) = \sum_{m\in[M]} \sum_{n\in[N]} O_n x_{m,n}(t) \tag{1}$$

[1]We use $[X]$ to represent the set $\{1, 2, \dots, X\}$ in the paper.

Authorized licensed use limited to: University of Wollongong. Downloaded on April 04,2020 at 13:46:08 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE I
NOTATION.

S	# of service chains
M	# of servers
T	# of time slots
N	# of VNFs
O_n	operating cost per instance of VNF n
D_n	deployment cost for deploying VNF n anew in a server
P_n	processing capability per instance of VNF n
C_k	capacity of resource k in each server
$c_{n,k}$	resource requirement for resource k per instance of VNF n
$\lambda_n^s(t)$	flow change ratio of VNF n a for service chain s at t
$\tilde{\lambda}_n^s(t)$	accumulated flow change ratio from first VNF in service chain s to VNF n at t
$f^s(t)$	flow rate of service chain s at t
$e_{n,n'}^s$	VNF n is the predecessor of VNF n' in service chain s
$u_n(t)$	minimal # of instances of VNF n at t
$x_{m,n}(t)$	# of VNF n deployed in m at t
$d_{m,n}(t)$	$[x_{m,n}(t) - x_{m,n}(t-1)]^+$
$y_{m,n,m',n'}^s(t)$	amount of traffic in service chain s forwarded from VNF n in server m to VNF n' in server m' in t

2) Deployment Costs: We use D_n to denote the cost for deploying VNF n anew in a server m, while there is no instance of VNF n deployed in m in the previous time slot. The cost is mainly due to the effort of copying the VNF's image to the server, and launching a VM/container with the image. The cost is typically considered on the order of the operating cost to run a server for a short period [20].

The total deployment cost in t can be formulated as:

$$C_{deploy}(t) = \sum_{m\in[M]} \sum_{n\in[N]} D_n[x_{m,n}(t) - x_{m,n}(t-1)]^+ \quad (2)$$

where $[x_{m,n}(t) - x_{m,n}(t-1)]^+ = \max\{0, x_{m,n}(t) - x_{m,n}(t-1)\}$.

The overall cost of the NFV system over the entire time span T is hence:

$$C_{all} = \sum_{t\in[T]} (C_{operate}(t) + C_{deploy}(t)) \quad (3)$$

We list important notation in this section in Table I.

The objective of our system is to minimize the overall cost, C_{all}. We can formulate the offline optimization problem as an integer linear programming (ILP) as follows:

$$\text{Minimize } C_{all} \quad (4)$$

Subject to :

$$P_n x_{m,n}(t) \geq \sum_{s\in[S]} \sum_{m'\in[M]\cup 0} \sum_{n'\in[N]\cup 0} e_{n',n}^s y_{m',n',m,n}^s(t), \quad \forall m \in [M], n \in [N], t \in [T] \quad (4a)$$

$$\sum_{n\in[N]} c_{n,k} x_{m,n}(t) \leq C_k, \quad \forall m \in [M], k \in \{0,1\}, t \in [T] \quad (4b)$$

$$\sum_{m\in[M]} \sum_{n\in[N]} e_{0,n}^s y_{0,0,m,n}^s(t) \geq f_s(t), \forall s \in [S], t \in [T] \quad (4c)$$

$$\sum_{m'\in[M]} \sum_{n'\in[N]} e_{n',n}^s y_{m',n',m,n}^s(t) = \lambda_n^s(t) \sum_{m'\in[M]} \sum_{n'\in[N]} e_{n,n'}^s y_{m,n,m',n'}^s(t), \quad \forall s \in [S], n \in [N], m \in [M], t \in [T] \quad (4d)$$

$$x_{m,n}(t) \in \{0,1,2,\ldots\}, \forall m \in [M], n \in [N], t \in [T] \quad (4e)$$

$$y_{m,n,m',n'}^s(t) \geq 0, \quad \forall m, m' \in [M], n, n' \in [N], s \in [S], t \in [T] \quad (4f)$$

Constraint (4a) ensure that NFV provider deploys enough VNF instances for all flows of all service chains at each time slot. Constraint (4b) makes sure that each resource required by VNF instances at each server does not exceed the resource capacity. In constraint (4c), $y_{0,0,m,n}^s(t)$ denotes the incoming traffic rate from the dummy VNF 0 on an imaginary server 0 directed to instances of VNF n on server m. Thus, constraint (4c) ensures the deployed VNFs handle all the incoming flow of the service chain. Constraint (4d) represents flow conservation with flow change ratio in consideration.

C. A Simplified Offline Optimization Problem

By carefully studying the structure of the offline optimization problem (4), we observe that the offline optimization problem can be simplified through removing routing variable $y_{m,n,m',n'}^s(t)$ and calculating the minimal number of instances for each VNF type in the entire system. We derive the minimal number of VNF instances based on constraints (4a), (4c) and (4d). In addition, we replace $[x_{m,n}(t) - x_{m,n}(t-1)]^+$ with variable $d_{m,n}(t)$.

Theorem 1: The minimal number of instances of VNF n for servicing all the incoming traffic across service chains at t is $u_n(t) = \left\lceil \frac{\sum_{s\in[S]} \tilde{\lambda}_n^s(t) f^s(t)}{P_n} \right\rceil$.

The detailed proof is given in Appendix A.

Therefore, we convert the original offline optimization problem (4) into the following simplified one:

$$\text{Minimize } \sum_{t\in[T]} \sum_{m\in[M]} \sum_{n\in[N]} (O_n x_{m,n}(t) + D_n d_{m,n}(t)) \quad (5)$$

Subject to:

$$\sum_{m\in[M]} x_{m,n}(t) \geq u_n(t), \quad \forall n \in [N], t \in [T] \quad (5a)$$

$$\sum_{n\in[N]} c_{n,k} x_{m,n}(t) \leq C_k, \quad \forall m \in [M], k \in \{0,1\}, t \in [T] \quad (5b)$$

$$x_{m,n}(t) - x_{m,n}(t-1) \leq d_{m,n}(t), \quad \forall m \in [M], n \in [N], t \in [T] \quad (5c)$$

$$d_{m,n}(t) \geq 0, \forall m \in [M], n \in [N], t \in [T] \quad (5d)$$

$$x_{m,n}(t) \in \{0,1,2,\ldots\}, \quad \forall m \in [M], n \in [N], t \in [T] \quad (5e)$$

Theorem 2: The above offline VNF provisioning problem in (5) is equivalent to the offline problem in (4).

The detailed proof is given in Appendix B

In the following sections, we relax the integrality constraints of (5) and apply a novel regularization method to design an efficient online algorithm for solving the relaxed

Authorized licensed use limited to: University of Wollongong. Downloaded on April 04,2020 at 13:46:08 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

linear program. Then we design efficient rounding algorithms for producing the integer solutions.

IV. Fractional Online Algorithm via Regularization

A. Regularization Method

Let P denote the relaxed LP of (5). The dual program [21] of P is donated by D. We derive D as follows:

$$\text{Maximize} \sum_{n\in[N]}\sum_{t\in[T]} u_n(t)a_n(t) - \sum_{m\in[M]}\sum_{k\in\{0,1\}}\sum_{t\in[T]} C_k b_{m,k}(t) \quad (6)$$

Subject to:

$$a_n(t) - \sum_{k\in\{0,1\}} c_{n,k}b_{m,k}(t) - w_{m,n}(t) + w_{m,n}(t+1) \le O_n, \quad \forall m\in[M], n\in[N], t\in[T] \quad (6a)(6b)$$

$$w_{m,n}(t) \le D_n, \quad \forall m\in[M], k\in\{0,1\}, t\in[T] \quad (6c)$$

$$a_n(t) \ge 0, \quad \forall n\in[N], t\in[T] \quad (6d)$$

$$b_{m,k}(t) \ge 0, \quad \forall m\in[m], k\in\{0,1\}, t\in[T] \quad (6e)$$

$$w_{m,n}(t) \ge 0, \quad \forall m\in[m], n\in[N], t\in[T] \quad (6f)$$

where, $a_n(t)$, $b_{m,k}(t)$ and $w_{m,n}(t)$ are Lagrangian dual variables associated with (5a), (5b), and (5c), respectively.

The main idea of our fractional online algorithm is to remove the correlation of our objective function between time slot $t-1$ and t, and decouple the original offline problem into multiple sub-problems, each of which is solvable at a time slot. To this end, we lift constraint (5c) into our objective function with a smooth convex function, $\Delta(x_{m,n}(t)||x_{m,n}(t-1))$, through regularization technique [2]:

$$x_{m,n}(t)\ln\frac{x_{m,n}(t)}{x_{m,n}(t-1)} + x_{m,n}(t-1) - x_{m,n}(t) \quad (7)$$

This smooth convex function is the sum of the relative entropy ($x_{m,n}(t)\ln\frac{x_{m,n}(t)}{x_{m,n}(t-1)}$) and the movement cost in linear term. To ensure the validity of our convex function, we need to guarantee denominator of $\frac{x_{m,n}(t)}{x_{m,n}(t-1)}$ is non-zero. Consequently, we add a positive constant term $\frac{\epsilon}{MN}$ to $x_{m,n}(t)$ and $x_{m,n}(t-1)$ in case $x_{m,n}(t-1)=0$. The positive constant value ϵ should be small enough, satisfying:

$$\epsilon \le \frac{MN}{e^{\frac{D_n}{O_n}\ln X_{max}} - 1}, \forall n\in[N] \quad (8)$$

where X_{max} denotes the maximum number of VNF instances of any type that can be deployed onto one server at the same time. Moreover, we define a weight parameter $\eta = \ln(1+\frac{MN}{\epsilon})$. We multiply (7) with $1/\eta$ to normalize the deployment cost by regularization.

Let $\widetilde{P}$ represent the new problem in which we lift constraint (5c) by adding the above constructed function into the objective function. Let $\widetilde{P}(t)$ denote the new sub-problem of $\widetilde{P}$ at t. Thus, we have $\widetilde{P} = \sum_{t\in[T]}\widetilde{P}(t)$. We present $\widetilde{P}(t)$ as follows:

$$\text{Minimize} \sum_{m\in[M]}\sum_{n\in[N]} O_n x_{m,n}(t) + \sum_{m\in[M]}\sum_{n\in[N]}\frac{D_n}{\eta} \times \left((x_{m,n}(t)+\frac{\epsilon}{MN})\ln\frac{x_{m,n}(t)+\frac{\epsilon}{MN}}{x_{m,n}(t-1)+\frac{\epsilon}{MN}} + x_{m,n}(t-1) - x_{m,n}(t)\right) \quad (9)$$

Algorithm 1 Online Regularization-Based Fractional Algorithm - ORFA

Input: $\mathbf{x}(t-1), M, N, \mathbf{O}, \mathbf{D}, \mathbf{C}, \mathbf{c}, \mathbf{u}, \epsilon$
Output: $\mathbf{x}^*(t)$
1: Set $\mathbf{x}=0$;
2: **for** $t\in[T]$ **do**
3: Observe $\mathbf{u}$;
4: Leverage *Interior Point Method* to solve $\widetilde{P}(t)$;
5: **return**: optimal solutions to $\widetilde{P}(t)$: **x(t)**;
6: **end for**

$$\text{Subject to:} \sum_{m\in[M]} x_{m,n}(t) \ge u_n(t), \quad \forall n\in[N] \quad (9a)$$

$$\sum_{n\in[N]} c_{n,k}x_{m,n}(t) \le C_k, \quad \forall m\in[M], k\in\{0,1\} \quad (9b)$$

$$x_{m,n}(t)\in\{0,1,2,\ldots\}, \quad \forall m\in[M], n\in[N] \quad (9c)$$

B. Online Algorithm

We present our online algorithm that derives a fractional solution to P in (5) in Alg. 1, ORFA, by solving $\widetilde{P}(t)$ at each time step t, respectively. Since $\widetilde{P}(t)$ is a standard convex problem, it can be optimally solved in polynomial time by the interior point method [21]. At each time slot t, Alg. 1 calculates an optimal solution to $\widetilde{P}(t)$ which is independent of the rounds prior to $t-1$. Based on Theorem 3, we observe that Alg. 1 produces a feasible solution to P, the original relaxed offline problem.

Theorem 3: Alg. 1, ORFA, produces a feasible solution of P in polynomial time.
The detailed proof is given in Appendix C.

C. Competitive Analysis of ORFA

We analyze the competitive ratio achieved by Alg. 1 through the primal-dual framework. We use P^* and D^* to denote the optimal value of P and D. In addition, we abuse P and D slightly to denote the objective value of P and D, respectively. Strong duality guarantees that $P^*=D^*$. Any feasible solution D serves as a lower bound to P^*. Therefore, the key point is to construct a feasible solution to D.

To derive a feasible solution to D, we explore $\widetilde{P}$ and its dual problem, $\widetilde{D}$. We define the Lagrangian dual variables in $\widetilde{D}$ as $\widetilde{a}_n(t)$ and $\widetilde{b}_{m,k}(t)$, associated with constraints (9a) and (9b), respectively. The optimal solution to $\widetilde{P}$ must satisfy the KKT condition [21]:

$$\text{KKT Optimality Condition} \quad (10)$$

presented in Table II.

Solving the KKT Optimality Condition, a solution to D can be derived:

$$a_n(t) = a_n^*(t), b_{m,k}(t) = b_{m,k}^*(t),$$
$$w_{m,n}(t) = \frac{D_n}{\eta}\ln\frac{1+\frac{\epsilon}{MN}}{x_{m,n}^*(t-1)+\frac{\epsilon}{MN}}$$

Theorem 4: The dual solution we obtain is a feasible solution to D.
The detailed proof is given in Appendix D.

Authorized licensed use limited to: University of Wollongong. Downloaded on April 04,2020 at 13:46:08 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE II
KKT OPTIMALITY CONDITION FOR $\widetilde{P}$ AND $\widetilde{D}$

$\forall n \in [N], t \in [T]:$
$\sum_{m\in[M]} x^*_{m,n}(t) - u_n(t) \geq 0$ (10.1)
$a^*_n(t)(\sum_{m\in[M]} x^*_{m,n}(t) - u_n(t)) = 0$ (10.2)
$\forall m \in [m], k \in [K], t \in [T]:$
$\sum_{n\in[N]} c_{n,k} x^*_{m,n}(t) - C_k \leq 0$ (10.3)
$b^*_{m,k}(t)(\sum_{n\in[N]} c_{n,k} x^*_{m,n}(t) - C_k) = 0$ (10.4)
$\forall m \in [m], n \in [N], t \in [T]:$
$O_n + \frac{D_n}{\eta} \ln \frac{x^*_{m,n}(t)+\frac{\epsilon}{MN}}{x^*_{m,n}(t-1)+\frac{\epsilon}{MN}} - a^*_n(t) +$ $\sum_{k\in[K]} c_{n,k} b^*_{m,k}(t) \geq 0$ (10.5)
$x^*_{m,n}(t) \times$ $(O_n + \frac{D_n}{\eta} \ln \frac{x^*_{m,n}(t)+\frac{\epsilon}{MN}}{x^*_{m,n}(t-1)+\frac{\epsilon}{MN}} - a^*_n(t) +$ $\sum_{k\in[K]} c_{n,k} b^*_{m,k}(t)) = 0$ (10.6)

Based on Theorem 4, we now compare the total cost using online algorithm ORFA with that of the offline optimal algorithm. We observe that our cost is comprised of operating costs and deployment costs. We derive the competitive ratio in terms of each cost to obtain the overall competitive ratio.

Lemma 1: The operating cost of P is no larger than D. The detailed proof is given in Appendix E.

Lemma 2: The deployment cost of P is no larger than $\ln(1+\frac{MN}{\epsilon})(1+\frac{\epsilon}{MN})$ times of D. The detailed proof is given in Appendix F.

Theorem 5: Our online algorithm ORFA achieves a $1+\ln(1+\frac{MN}{\epsilon})(1+\frac{\epsilon}{MN})$-ratio compared to the offline optimum, P^*, of the original problem in (5).

Proof: Taking the ratios in Lemma (1) and (2), we have:

$$\frac{P(OFRA)}{D} \leq 1 + ln(1+\frac{MN}{\epsilon})(1+\frac{\epsilon}{MN})$$

Thus, we have:

$$\begin{aligned} P(OFRA) &\leq (1 + ln(1+\frac{MN}{\epsilon})(1+\frac{\epsilon}{MN}))D \\ &\leq (1 + ln(1+\frac{MN}{\epsilon})(1+\frac{\epsilon}{MN}))P^* \end{aligned}$$

□

V. ROUNDING ALGORITHM BASED ON LINEAR ALGEBRA AND RANDOMIZATION

A. Rounding Scheme

We now present a rounding algorithm, RA, that rounds the fractional solution $x^*_{m,n}(t)$ obtained by Alg. 1 in Sec. IV to an integral solution denoted by $\bar{x}_{m,n}(t)$. We design the rounding algorithm based on [3].

Our rounding algorithm is composed of two phases: **Phase 1** applies a linear-algebra-based rounding technique that reduces the number of fractional variables to construct a simple bipartite graph, and **Phase 2** leverages dependent rounding technique [22] to compute the final solution. Each iteration h is in either **Phase 1** or **Phase 2**. The value of $x_{m,n}(t)$ after iteration h is denoted by $x^h_{m,n}(t)$; the fractional part of $x^h_{m,n}(t)$ is represented as $\rho^h_{m,n}(t)$, *i.e.*, $\rho^h_{m,n}(t) = x^h_{m,n}(t) - \lfloor x^h_{m,n}(t) \rfloor$.

The algorithm, RA, starts from **Phase 1** and then goes into **Phase 2**. At the beginning of iteration $h+1$, let a VNF n be called a floating VNF, if any instance of VNF n is currently assigned fractionally to more than one server, *i.e.*, $\rho^h_{m,n}(t) \in (0,1)$ for more than one m. We call a server m a floating server if it currently has at least one floating VNF assigned to it. In addition, a server is a **key** server if it is assigned with no less than **four** floating VNFs. We use N', M_f and M' to represent the set of floating VNFs, floating servers and **key** servers at the current iteration. The set of currently unrounded pairs is expressed as V, *i.e.*, $V = \{(m,n) : \rho^h_{m,n}(t) \in (0,1)\}$. The current set of unrounded pairs between floating VNFs and **key** servers is denoted as V', *i.e.*, $V' = \{(m,n) : \rho^h_{m,n}(t) \in (0,1), m \in M', n \in N'\}$.

Now we discuss the two phases in detail.

Phase 1: The current iteration is in **Phase 1** if $|V'| > |N'| + 2|M'|$. We consider the following linear system:

$$\sum_{m\in[M']} x_{m,n}(t) = \sum_{m\in[M']} x^h_{m,n}(t), \quad \forall n \in N' \tag{11}$$

$$\sum_{n\in N'} c_{n,k} x_{m,n}(t) = \sum_{n\in N'} c_{n,k} x^h_{m,n}(t), \quad \forall m \in M', k \in \{0,1\} \tag{12}$$

We use $Ax = b$ to represent the above linear system. By ensuring $|V'| > |N'| + 2|M'|$, we observe that the number of variables, $|V'|$, exceeds the number of constraints. Thus, we can find in polynomial time a non-zero vector r that satisfies $Ar = 0$. Furthermore, we calculate two positive values α and β that satisfy:

- All entries of $\rho + \alpha r$ and $\rho - \beta r$ are in [0,1];
- At least one entry in $\rho + \alpha r$ and $\rho - \beta r$ is in {0,1}.

Eventually, with probability $\frac{\beta}{\beta+\alpha}$, RA asets $x^{h+1}_{m,n}(t) = x^h_{m,n}(t) + \alpha r, \forall (m,n) \in V'$; with probability $\frac{\alpha}{\beta+\alpha}$, RA sets $x^{h+1}_{m,n}(t) = x^h_{m,n}(t) - \beta r, \forall (m,n) \in V'$. $\forall (m,n) \notin V$, $x^{h+1}_{m,n}(t) = x^h_{m,n}(t)$.

We observe that while $|V'| > |N'| + 2|M'|$, we can find such non-zero vector r and perform the above rounding process accordingly. During the process, the numbers of unrounded pairs, floating VNFs and key servers decrease. Once we encounter $|V'| \leq |N'| + 2|M'|$, RA transits into **Phase 2**. Otherwise, RA stays in **Phase 1** until all the fractional assignments are eliminated.

Phase 2: If the current iteration is in **Phase 2**, we construct a bipartite graph $G = (M_f, N', E)$, where we set up an edge (m,n) iff $(m,n) \in V$. RA leverages Depth-First-Search to find an even circle[2] or a maximal path[3] in G, and partitions the edges in it into two distinct matchings Ψ_1 and Ψ_2, *e.g.* if (m,n) is in Ψ_1, than the next edge (m',n) must be in Ψ_2.

Define

$$\begin{aligned} \alpha = min\{\gamma > 0 : ((\exists (m,n) \in \Psi_1 : \rho^h_{m,n}(t) + \gamma = 1) \\ \vee (\exists (m,n) \in \Psi_2 : \rho^h_{m,n}(t) - \gamma = 0))\} \\ \beta = min\{\gamma > 0 : ((\exists (m,n) \in \Psi_1 : \rho^h_{m,n}(t) - \gamma = 0) \\ \vee (\exists (m,n) \in \Psi_2 : \rho^h_{m,n}(t) + \gamma = 1))\} \end{aligned}$$

[2]An even circle is a circle containing an even number of edges.
[3]A maximal path is a non-circle path with the maximal number of edges

Authorized licensed use limited to: University of Wollongong. Downloaded on April 04,2020 at 13:46:08 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Algorithm 2 Rounding Algorithm - RA

Input: $\mathbf{x}^*(t), M, N, \mathbf{O}, \mathbf{D}, \mathbf{C}, \mathbf{c}$
Output: $\bar{\mathbf{x}}(t)$

Set $\mathbf{x} = \mathbf{x}^*(t)$
Set $phase = 1$;
while $V \neq \varnothing$ **do**
  if $|V'| \leq |N'| + 2|M'|$ and $phase == 1$ **then**
    $phase = 2$;
  end if
  if $phase == 1$ **then**
    Obtain the non-zero vector r based on linear system (11) and (12);
    Calculate α and β according to r;
    With probability $\frac{\beta}{\beta+\alpha}$, set $x_{m,n}(t) = x_{m,n}(t) + \alpha r, \forall (m,n) \in V$;
    With probability $\frac{\alpha}{\beta+\alpha}$, sets $x_{m,n}(t) = x_{m,n}(t) - \beta r, \forall (m,n) \in V$;
    Removing (m,n) from V if $x_{m,n}(t)$ is integer;
  else
    Set $phase$ = 2;
    Construct the bipartite graph $G = (M_f, N', E)$;
    Leverage depth-first-search algorithm to find the maximum path / a even circle in G and partition the edges in it into to matchings, Ψ_1 and Ψ_2;
    Set $\rho_{m,n}(t) = x_{m,n}(t) - \lfloor x_{m,n}(t) \rfloor, \forall (m,n) \in V$;
    Define

$$\alpha = min\{\gamma > 0 : ((\exists (m,n) \in \Psi_1 : \rho_{m,n}(t) + \gamma = 1) \vee (\exists (m,n) \in \Psi_2 : \rho_{m,n}(t) - \gamma = 0))\}$$
$$\beta = min\{\gamma > 0 : ((\exists (m,n) \in \Psi_1 : \rho_{m,n}(t) - \gamma = 0) \vee (\exists (m,n) \in \Psi_2 : \rho_{m,n}(t) + \gamma = 1))\}$$

    With probability $\frac{\beta}{\beta+\alpha}$, set $x_{m,n}(t) = x_{m,n}(t) + \alpha, \forall (m,n) \in \Psi_1$, and $x_{m,n}(t) = x_{m,n}(t) - \alpha, \forall (m,n) \in \Psi_2$;
    With probability $\frac{\alpha}{\beta+\alpha}$, set $x_{m,n}(t) = x_{m,n}(t) - \beta, \forall (m,n) \in \Psi_1$, and $x_{m,n}(t) = x_{m,n}(t) + \beta, \forall (m,n) \in \Psi_2$.
    Removing (m,n) from V if $x_{m,n}(t)$ is integer;
  end if
end while
Set $\bar{\mathbf{x}}(t) = \mathbf{x}$

The rounding step is as follows:

With probability $\frac{\beta}{\beta+\alpha}$, set $x^{h+1}_{m,n}(t) = x^h_{m,n}(t) + \alpha, \forall (m,n) \in \Psi_1$, and $x^{h+1}_{m,n}(t) = x^h_{m,n}(t) - \alpha, \forall (m,n) \in \Psi_2$;

With probability $\frac{\alpha}{\beta+\alpha}$, set $x^{h+1}_{m,n}(t) = x^h_{m,n}(t) - \beta, \forall (m,n) \in \Psi_1$, and $x^{h+1}_{m,n}(t) = x^h_{m,n}(t) + \beta, \forall (m,n) \in \Psi_2$.

We describe the complete rounding algorithm, RA, in Alg. 2.

VI. The Complete Online Algorithm

We present the complete online algorithm in Alg. 3.

A. Competitive Analysis of COA

Now we analyze the competitive ratio obtained by Alg. 3.

Algorithm 3 Complete Online Algorithm - COA

Input: $M, N, \mathbf{O}, \mathbf{D}, \mathbf{C}, \mathbf{c}$
Output: $\bar{\mathbf{x}}$

Set $\bar{\mathbf{x}}(0) = \mathbf{0}$;
Initialize ϵ;
for $t \in [T]$ **do**
  $\mathbf{x}^*(t)$ = ORFA($\bar{\mathbf{x}}(t-1), M, N, \mathbf{O}, \mathbf{D}, \mathbf{C}, \mathbf{c}, \mathbf{u}, \epsilon$);
  $\bar{\mathbf{x}}(t)$ = RA($\mathbf{x}^*(t), M, N, \mathbf{O}, \mathbf{D}, \mathbf{C}, \mathbf{c}$);
end for

Lemma 3: RA ensures $\boldsymbol{E}[\bar{x}_{m,n}(t)] = x^*_{m,n}(t), \forall m \in [M], n \in [N], t \in [T]$.

The detailed proof is given in Appendix G.

Let $\bar{P}(COA)$ denote the objective value of original problem (5) obtained by Alg. 3. In addition, the operational costs and deployment cost through Alg. 3 are expressed as $\bar{P}(operate)$ and $\bar{P}(deploy)$, respectively.

Lemma 4: $\boldsymbol{E}[\bar{P}(operate)]$ *is no larger than* $P(OFRA)$.

The detailed proof is given in Appendix H.

Lemma 5: $\boldsymbol{E}[\bar{P}(deploy)]$ *is no larger than* Ω *times of* $P(OFRA)$, *where* $\Omega = \max_{n\in[N]}\{\frac{D_n}{O_n}\}$.

The detailed proof is given in Appendix I.

Theorem 6: $\boldsymbol{E}[\bar{P}(COA)]$ *is no larger than* $(1 + \ln(1 + \frac{MN}{\epsilon})(1 + \frac{\epsilon}{MN}))(1 + \Omega)$ *times the offline optimal objective value* P^* *of the original problem (5).*

Proof:

$$\begin{aligned}\mathbf{E}[\bar{P}(COA)] &= \mathbf{E}[\bar{P}(operate)] + \mathbf{E}[\bar{P}(deploy)] \\ &\leq (1+\Omega)P(OFRA) \\ &\leq (1+\ln(1+\frac{MN}{\epsilon})(1+\frac{\epsilon}{MN}))(1+\Omega)P^*\end{aligned}$$

□

B. Constraint Violation Analysis of COA

In the analysis of constraint violation, we make the following assumption:

Assumption 1:

$$\frac{O_n}{D_n} \geq \ln\left(\frac{X_{max}MN}{\epsilon} + 1\right) / \ln\left(\frac{MN}{\epsilon} + 1\right), \quad \forall n \in [N]$$

Typical $\frac{O_n}{D_n}$ is within $[5, 10]$ since deployment cost is on the order of the cost to run a VNF instance for several minutes [20]. Assuming $\frac{O_n}{D_n}$ is 5 and X_{max} is at most 20, we can safely set $\frac{\epsilon}{MN}$ to 0.1 to satisfy the above assumption. Assumption 1 ensures the objective function (9) is monotonically non-decreasing and paves the way for Lemma 6.

Lemma 6: Every VNF that needs to be rounded is assigned fractionally to more than one server.

The detailed proof is given in Appendix J.

Lemma 7: In any iteration of ***Phase 2****, any floating server has at most four floating VNFs assigned fractionally to it.*

The detailed proof is given in Appendix K.

Lemma 8: Suppose the first iteration entering ***Phase 2*** *is h. Our complete online algorithm, Alg. 3, ensures*:

$$\sum_{n\in N'} \bar{\rho}_{m,n}(t) \in \{\lfloor \sum_{n\in N'} \rho^h_{m,n}(t) \rfloor, \lceil \sum_{n\in N'} \rho^h_{m,n}(t) \rceil\}, \ \forall m \in [M]$$
$$\sum_{m\in M'} \bar{\rho}_{m,n}(t) \in \{\lfloor \sum_{m\in M'} \rho^h_{m,n}(t) \rfloor, \lceil \sum_{m\in M'} \rho^h_{m,n}(t) \rceil\}, \ \forall n \in [N]$$

Authorized licensed use limited to: University of Wollongong. Downloaded on April 04,2020 at 13:46:08 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE III
VNF CONFIGURATIONS

VNF	vCPU	Mem	Instance	Capacity Type	Change Ratio
NAT	2	8GB	m4.large	900Mbps	1.0
Firewall	4	16GB	m4.xlarge	900Mbps	0.8-1.0
Proxy	4	16GB	m4.xlarge	900Mbps	1.0
IDS	8	32GB	m4.2xlarge	600Mbps	0.8-1.0

where $\bar{\rho}_{m,n}(t)$ denotes the final rounding result of $\rho^h_{m,n}(t)$ produced by RA.

The detailed proof is given in Appendix L.

Theorem 7: For every floating VNF, we have with probability exactly 1:

$$\sum_{m\in[M]} \bar{x}_{m,n}(t) = u_n(t)$$

The detailed proof is given in Appendix M.

Theorem 8: For every floating server, we have with probability exactly 1:

$$\sum_{n\in[N]} c_{n,k}\bar{x}_{m,n}(t) \leq C_k + 2\max_{n\in[N]} c_{n,k} - \min_{n\in[N]} c_{n,k}, \forall k \in \{0,1\}$$

The detailed proof is given in Appendix N.

In conclusion, Alg. 3 ensures enough VNF instances for flow processing, with the resource capacity constraints violated by at most $2\max_{n\in[N],k\in\{0,1\}} c_{n,k} - \min_{n\in[N],k\in\{0,1\}} c_{n,k}$.

VII. PERFORMANCE EVALUATION

A. Simulation Setup

System Settings: We evaluate our online algorithm, Alg. 3, through trace-driven simulation. We set the whole period to be five days and each time slot lasts one hour, *i.e.* 120 time slots in total. Traffic rates to the service chain are generated based on real-world traffic statistics from Huawei Inc. The peak load is 720Mpbs appearing between 9 and 11 PM, and the peak-to-mean ratio (PMR) is 1.56.

Following the CPU requirements and process capacity presented in [23], we summaries the VNF configurations in Table III. We identify CPU and memory as the key resources. In addition, we set the memory requirements according to the respective VM instances provided by Amazon EC2 [24]. Change ratios for Firewall and Intrusion Detection System (IDS) are between 0.8 and 1, and change ratios for Network Address Translation (NAT) and Proxy remain 1. The operating costs per hour are equal to the Amazon on-demand prices for respective VM instances [25]. As the deployment cost is considered on the order of the operating cost to run a server for several minutes [20] and one time slot is one hour, we set $\frac{O_n}{D_n}$ to be 10.

The number of servers is set to 200. According to Amazon EC2, each vCPU is a thread of an Intel Xeon core and each core holds at most two threads [24]. Thus, we set the vCPU capacity of a server to be 2×24 units and the memory capacity to be 256GB. The base setting is 100 service chains, each containing 2-4 VNFs randomly chosen from the four VNF types. Parameter $\frac{\epsilon}{MN}$ is fixed to 0.1.

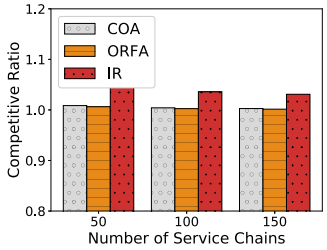

Fig. 1. Competitive ratio: different numbers of service chains.

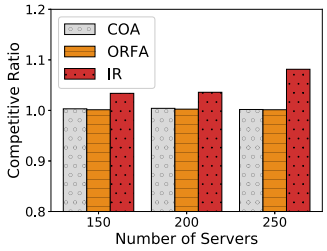

Fig. 2. Competitive ratio: different numbers of servers.

B. Performance of COA

We evaluate the *competitive ratio* achieves by COA, which is the ratio between the overall cost by COA and the offline optimal cost computed exactly by the CVXPY optimizer using Python. The results presented in Fig. 1-5 are obtained in scenarios without resource over-utilization.

Baselines: We propose the following baselines for comparison:

- ORFA: Performance of using fractional solution obtained by ORFA.
- IR: Performance of using a randomized independent rounding algorithm that rounds the fractional solution of ORFA to its nearest integral solution. The performance is counted only if the rounded solution is feasible.

Comparasion: We compare the competitive ratio achieved by COA with the baselines in different settings. We observe COA presents stable and near-optimal performance across various scenarios.

1) Different Numbers of Service Chains: Intuitively, a larger number of service chains results in a large amount of network traffic, which needs more VNF instances to process. Besides, the proportion of VNFs will also change more dramatically, yielding possible huge deployment costs when poorly scheduled. However, as Fig. 1 illustrated, COA exhibits stable performance as we vary the number of service chains from 50 to 100. COA achieves competitive ratios close to those by ORFA and significantly outperforms IR. Competitive ratios by COA are nearly 1 and are notably less than the theoretical expected upper bound which is 4.00 in our setting.

2) Different Numbers of Servers: We evaluate the impact of different server numbers on COA performance in Fig. 2. As the number of servers decreases, re-scheduling of computational resources for current VNF instances becomes more frequent. Again, we observe that COA handles the situation well with satisfying competitive ratios that are close to 1.

3) Different D_n/O_n: We further investigate how different ratios of deployment costs to operating costs affect the

Authorized licensed use limited to: University of Wollongong. Downloaded on April 04,2020 at 13:46:08 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

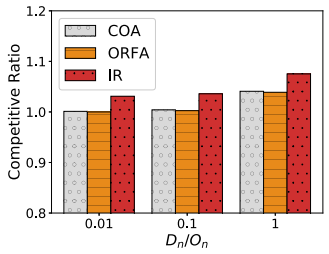

Fig. 3. Competitive ratio: different $\frac{D_n}{O_n}$.

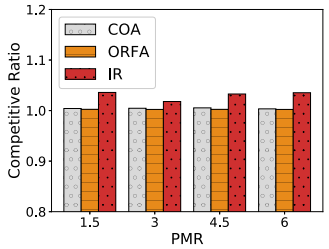

Fig. 4. Competitive ratio: different PMR.

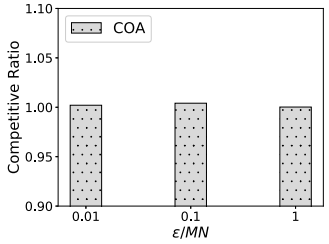

Fig. 5. Competitive ratio: different $\frac{\epsilon}{MN}$.

performance of COA in Fig. 3. In accordance with the theoretical analysis, we observe an apparent upward trend in competitive ratio as $\frac{D_n}{O_n}$ grows. However, simulation results demonstrate huge gaps between the competitive ratios and the theoretical upper bounds. The gaps are due to that the theoretical upper bounds only happen in worst cases which are rare in realistic scenarios.

4) Different PMR: Fig. 4 shows the performance of COA under different PMR. We fix the peak flow rate and reduce the mean flow rate to simulate different PMRs. A larger PMR implies a more dramatic traffic fluctuation in the NFV system. Nonetheless, our COA performs properly under different levels of PMR.

5) Different $\epsilon/(MN)$: Theorem 6 shows that $\frac{\epsilon}{MN}$ influences the theoretical upper bound. Therefore, we evaluate the performance of COA with different $\frac{\epsilon}{MN}$ in Fig. 5. We observe no obvious change in competitive ratio as we vary $\frac{\epsilon}{MN}$ from 0.01 to 1.

Over-Utilizing Resources: We are now turning our attention to over-utilizing resources. We fix the number of servers to 100 while varying the number of flows from 200 to 800. The results are summarized in Fig. 6 and Fig. 7.

Fig. 6 presents the percentage of resource-overuse time slots. We can observe an obvious upward trend as incoming flow traffic raises. This is consistent with our intuition: as the flow traffic becomes larger, the probability for resource overuse increases. However, the simulation results show that

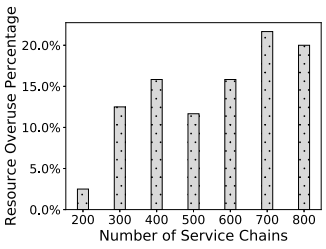

Fig. 6. Percentage: time slots with resource overuse.

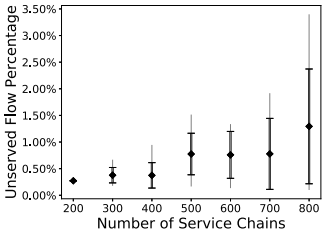

Fig. 7. Percentage: unserved flow rate.

TABLE IV
BACKUP SERVERS

# of Chains	# of Backup Servers	# of Chains	# of Backup Servers
100	0	500	2
200	1	600	2
300	1	700	2
400	1	800	3

most of the time slots do not suffer from violation of resource capacity. Moreover, most of the overuse situations occur during peak hours.

During the experiment, we observe that CPU is the bottleneck for our NFV system. However, the maximum CPU violation on a single server is at most 8 vCPUs, while our theoretical upper bound for CPU violation is 14 vCPUs. To tackle such resource over-utilization, our system removes some VNF instances to meet the resource constraints at the price of allowing some network flows to be unserved. Fig. 7 demonstrates the percentage of unserved flow rate. A black dot represents the average percentage; the thick lines indicate the standard errors of the percentage across the respective group of service chains, while the thinner grey lines show the maximum and minimum unserved flow percentages across these chains. We see that the percentage of unserved flow rate is very small. When the number of service chains is 800, the unserved percentage in a single time slot is at most 3.40%, and the average unserved percentage remains 1.29%. Thus, we can effectively solve such a dilemma by adding a small amount of back-up resources catering for the unserved network traffic, i.e., deploying additional VNF instances on backup servers. The numbers of backup servers needed at different total numbers of service chains are presented in Table IV. We observe that deploying three backup servers is sufficient to handle unserved flows from up to 800 service chains.

VIII. CONCLUSION

While the fast development of NFV enables flexible network function deployment, new challenges are introduced that require a more dynamic algorithm for VNF scaling.

Authorized licensed use limited to: University of Wollongong. Downloaded on April 04,2020 at 13:46:08 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Our work targets online VNF scaling in a cloud data center under multi-resource constraints. We present a novel online VNF scaling algorithm based on the regularization technique and dependent rounding. Our approach achieves upper-bounded competitive ratio and resource capacity constraint violation, according to thorough theoretical analysis. Trace-driven simulation further verifies the analytical results and demonstrates good performance of our method.

APPENDIX A
PROOF OF THEOREM 1

Due to the flow conservation constraints (4c) and (4d), the total incoming flow rate to all VNF n instances is $\sum_{s\in[S]} \bar{\lambda}_n^s(t) f^s(t)$. Since the process capability of one VNF n instance is P_n, the minimal instance number required is $\left\lceil \frac{\sum_{s\in[S]} \bar{\lambda}_n^s(t) f^s(t)}{P_n} \right\rceil$.

APPENDIX B
PROOF OF THEOREM 2

We derive constraint (5a) from constraints (4a), (4c) and (4d). Therefore, any feasible solution to problem (4) is a feasible solution to problem (5). On the other hand, given any feasible solution to problem (5), we can route total network flow to each type of VNF proportion to the number of VNF instances to each server. In these way, we can guarantee the feasibility of constraints (4a), (4c) and (4d) and obtain a feasible solution to problem (4). In conclusion, problem (4) and problem (5) share the same objective function and any feasible solution to one problem corresponds to one feasible solution to the other problem. Therefore, the two problem is equivalent.

APPENDIX C
PROOF OF THEOREM 3

Proof: $\widetilde{P}(t)$ is solved through *Interior Point Method* in polynomial time [21]. We observe that constraints (9a) and (9b) in $\widetilde{P}(t)$ equal to (5a) and (5b) in P. In addition, constraints (5c) in P can be easily met by calculate $d_{m,n}(t)$ based on $x_{m,n}(t)$. Thus, we show that each solution to $\widetilde{P}$ is a feasible solution to P. □

APPENDIX D
PROOF OF THEOREM 4

Proof: For (6b), we have:

$$\begin{aligned} w_{m,n}(t+1) - w_{m,n}(t) &= -\frac{D_n}{\eta} \ln \frac{x_{m,n}(t) + \frac{\epsilon}{MN}}{x_{m,n}(t-1) + \frac{\epsilon}{MN}} \\ &\le O_n - a_n(t) + \sum_{k\in\{0,1\}} c_{n,k} b_{m,k}(t) \end{aligned}$$

where the first inequality is due to (10.5).

For (6c), we have:

$$\begin{aligned} w_{m,n}(t) &= \frac{D_n}{\eta} \ln \frac{1 + \frac{\epsilon}{MN}}{x^*_{m,n}(t-1) + \frac{\epsilon}{MN}} \\ &\le D_n \end{aligned}$$

since $x^*_{m,n}(t-1) \ge 0$.

Thus, we show that the dual variables satisfy D. □

APPENDIX E
PROOF OF LEMMA 1

Proof:

$$\begin{aligned} \sum_{t\in[T]} C_{operate}(t) &= \sum_{t\in[T]} \sum_{m\in[M]} \sum_{n\in[N]} O_n x_{m,n}(t) \\ &= \sum_{t\in[T]} (\sum_{m\in[M]} \sum_{n\in[N]} a_n(t) x_{m,n}(t) \\ &\quad - \sum_{m\in[M]} \sum_{n\in[N]} \sum_{k\in\{0,1\}} c_{n,k} b_{m,k}(t) x_{m,n}(t) \\ &\quad - \sum_{m\in[M]} \sum_{n\in[N]} \frac{D_n x_{m,n}(t)}{\eta} \\ &\quad \times \ln \frac{x_{m,n}(t) + \frac{\epsilon}{MN}}{x_{m,n}(t-1) + \frac{\epsilon}{MN}}) \end{aligned} \tag{13}$$

Equality (13) follows from condition (10.6). Due to conditions (10.2) and (10.4), we ensure:

$$\begin{aligned} &\sum_{t\in[T]} \sum_{m\in[M]} \sum_{n\in[N]} a_n(t) x_{m,n}(t) \\ &\quad = \sum_{t\in[T]} \sum_{n\in[N]} u_n(t) a_n(t) \end{aligned} \tag{14}$$

$$\begin{aligned} &\sum_{t\in[T]} \sum_{m\in[M]} \sum_{n\in[N]} \sum_{k\in\{0,1\}} c_{n,k} b_{m,k}(t) x_{m,n}(t) \\ &\quad = \sum_{t\in[T]} \sum_{m\in[M]} \sum_{k\in\{0,1\}} C_k b_{m,k}(t) \end{aligned} \tag{15}$$

Furthermore, we have:

$$\begin{aligned} &\sum_{t\in[T]} \sum_{m\in[M]} \sum_{n\in[N]} \frac{D_n x_{m,n}(t)}{\eta} \ln \frac{x_{m,n}(t) + \frac{\epsilon}{MN}}{x_{m,n}(t-1) + \frac{\epsilon}{MN}} \\ &\quad = \sum_{m\in[M]} \sum_{n\in[N]} \frac{D_n}{\eta} (\sum_{t\in[T]} (x_{m,n}(t) + \frac{\epsilon}{MN}) \\ &\qquad \times \ln \frac{x_{m,n}(t) + \frac{\epsilon}{MN}}{x_{m,n}(t-1) + \frac{\epsilon}{MN}} \\ &\qquad - \sum_{t\in[T]} \frac{\epsilon}{MN} \ln \frac{x_{m,n}(t) + \frac{\epsilon}{MN}}{x_{m,n}(t-1) + \frac{\epsilon}{MN}}) \end{aligned} \tag{16}$$

Following by telescopic sum and the fact that:

$$\sum_i a_i \log(\frac{a_i}{b_i}) \ge (\sum_i a_i) \log(\frac{\sum_i a_i}{\sum_i b_i})$$

and:

$$a - b \le a \ln(a/b)$$

Since $x_{m,n}(0) = 0$, We have:

$$\begin{aligned} &- \sum_{t\in[T]} \frac{\epsilon}{MN} \ln \frac{x_{m,n}(t) + \frac{\epsilon}{MN}}{x_{m,n}(t-1) + \frac{\epsilon}{MN}} \\ &\quad = -\frac{\epsilon}{MN} \ln \frac{x_{m,n}(T) + \frac{\epsilon}{MN}}{x_{m,n}(0) + \frac{\epsilon}{MN}} \\ &\quad \ge -\frac{\epsilon}{MN} \ln \frac{x_{m,n}(T) + \frac{\epsilon}{MN}}{x_{m,n}(0) + \frac{\epsilon}{MN}} \\ &\quad = -(x_{m,n}(0) + \frac{\epsilon}{MN}) \ln \frac{x_{m,n}(T) + \frac{\epsilon}{MN}}{x_{m,n}(0) + \frac{\epsilon}{MN}} \\ &\quad \ge x_{m,n}(0) - x_{m,n}(T) \end{aligned} \tag{17}$$

Authorized licensed use limited to: University of Wollongong. Downloaded on April 04,2020 at 13:46:08 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

and:

$$\begin{aligned}
&\sum_{t\in[T]}(x_{m,n}(t)+\frac{\epsilon}{MN})\ln\frac{x_{m,n}(t)+\frac{\epsilon}{MN}}{x_{m,n}(t-1)+\frac{\epsilon}{MN}}\\
&\geq(\sum_{t\in[T]}(x_{m,n}(t)+\frac{\epsilon}{MN}))\ln\frac{\sum_{t\in[T]}(x_{m,n}(t)+\frac{\epsilon}{MN})}{\sum_{t\in[T]}(x_{m,n}(t-1)+\frac{\epsilon}{MN})}\\
&\geq\sum_{t\in[T]}(x_{m,n}(t)+\frac{\epsilon}{MN})-\sum_{t\in[T]}(x_{m,n}(t-1)+\frac{\epsilon}{MN})\\
&=x_{m,n}(T)-x_{m,n}(0) \qquad (18)
\end{aligned}$$

Plug (17) and (18) into (13), we arrive at the result that:

$$\begin{aligned}
&\sum_{t\in[T]}C_{operate}(t)\\
&\leq\sum_{t\in[T]}\sum_{n\in[N]}u_n(t)a_n(t)-\sum_{t\in[T]}\sum_{m\in[M]}\sum_{k\in\{0,1\}}C_kb_{m,k}(t)\\
&=\text{value of } D \qquad (19)
\end{aligned}$$

Therefore, we prove that operating cost in P is no larger than D. □

APPENDIX F
PROOF OF LEMMA 2

Proof:

$$\begin{aligned}
&C_{deploy}(t)\\
&=\eta\sum_{m\in[M]}\sum_{n\in[N]}\sum_{x_{m,n}(t)>x_{m,n}(t-1)}\\
&\quad\frac{D_n}{\eta}(x_{m,n}(t)-x_{m,n}(t-1))\\
&\leq\eta\sum_{m\in[M]}\sum_{n\in[N]}\sum_{x_{m,n}(t)>x_{m,n}(t-1)}\\
&\quad((x_{m,n}(t)+\frac{\epsilon}{MN})\frac{D_n}{\eta}\ln\frac{x_{m,n}(t)+\frac{\epsilon}{MN}}{x_{m,n}(t-1)+\frac{\epsilon}{MN}} \qquad (20)\\
&=\eta\sum_{m\in[M]}\sum_{n\in[N]}\sum_{x_{m,n}(t)>x_{m,n}(t-1)}((1+\frac{\epsilon}{MNx_{m,n}(t)})\\
&\quad x_{m,n}(t)(-O_n+a_n(t)-\sum_{k\in\{0,1\}}c_{n,k}b_{m,k}(t)) \qquad (21)\\
&\leq\eta(1+\frac{\epsilon}{MN})\sum_{m\in[M]}\sum_{n\in[N]}\sum_{x_{m,n}(t)>x_{m,n}(t-1)}\\
&\quad(x_{m,n}(t)(-O_n+a_n(t)-\sum_{k\in\{0,1\}}c_{n,k}b_{m,k}(t)) \qquad (22)\\
&\leq\eta(1+\frac{\epsilon}{MN})\sum_{m\in[M]}\sum_{n\in[N]}\\
&\quad(x_{m,n}(t)(a_n(t)-\sum_{k\in\{0,1\}}c_{n,k}b_{m,k}(t))) \qquad (23)\\
&=\eta(1+\frac{\epsilon}{MN})\sum_{m\in[M]}\sum_{n\in[N]}\\
&\quad(x_{m,n}(t)a_n(t)-\sum_{k\in\{0,1\}}x_{m,n}(t)c_{n,k}b_{m,k}(t)) \qquad (24)
\end{aligned}$$

Inequality (20) follows as $a-b\leq a\ln(a/b)$. Equality (21) follows from condition (10.6) as $x_{m,n}(t)>x_{m,n}(t-1)\geq 0$. Inequality (22) follows as $x_{m,n}(t-1)\geq 1$. To prove inequality (23), we consider other three conditions:

- $x_{m,n}(t)=x_{m,n}(t-1)$

  In these case, we have:

$$\begin{aligned}
-O_n+a_n(t)-\sum_{k\in\{0,1\}}c_{n,k}b_{m,k}(t)&=\frac{D_n}{\eta}\ln\frac{x_{m,n}(t)+\frac{\epsilon}{MN}}{x_{m,n}(t-1)+\frac{\epsilon}{MN}}\\
&=0
\end{aligned}$$

  Since $O_n>0$, we have $a_n(t)-\sum_{k\in\{0,1\}}c_{n,k}b_{m,k}(t)>0$

- $x_{m,n}(t)<x_{m,n}(t-1)$ and $x_{m,n}(t)=0$.

  Then $x_{m,n}(t)(a_n(t)-\sum_{k\in\{0,1\}}c_{n,k}b_{m,k}(t))=0$.

- $x_{m,n}(t)<x_{m,n}(t-1)$ and $x_{m,n}(t)>0$.

  We have:

$$\begin{aligned}
&a_n(t)-\sum_{k\in\{0,1\}}c_{n,k}b_{m,k}(t)\\
&=O_n+\frac{D_n}{\eta}\ln\frac{x_{m,n}(t)+\frac{\epsilon}{MN}}{x_{m,n}(t-1)+\frac{\epsilon}{MN}}\\
&\geq O_n+\frac{D_n}{\eta}\ln\frac{x_{m,n}(t)}{x_{m,n}(t-1)} \qquad (25)\\
&\geq O_n+\frac{D_n}{\eta}\ln\frac{1}{X_{max}} \qquad (26)\\
&=O_n-\frac{D_n}{\ln(1+\frac{MN}{\epsilon})}\ln X_{max}\\
&\geq O_n-\frac{D_n}{\ln(1+\frac{\frac{MN}{MN}}{e^{\frac{D_n}{O_n}\ln X_{max}}-1})}\\
&\quad\times\ln X_{max} \qquad (27)\\
&=0
\end{aligned}$$

Inequality (25) holds as $\frac{a+c}{b+c}\geq\frac{a}{b}, b>a>0, c>0$. Inequality (26) follows when $x_{m,n}(t)=1$ and $x_{m,n}(t-1)=X_{max}$. Inequality (27) follows as we substitute ϵ with its upper-bound.

Consider all three above conditions, we show inequality (23) holds.

Now we first consider term $\sum_{m\in[M]}\sum_{n\in[N]}x_{m,n}(t)a_n(t)$. If $a_n(t)=0$, we have:

$$\sum_{m\in[M]}x_{m,n}(t)a_n(t)=u_n(t)a_n(t)$$

And if $a_n(t)>0$, by condition (10.2), we ensure $\sum_{m\in[M]}x_{m,n}(t)=u_n(t),\forall n\in[N]$. Therefore, we obtain that:

$$\sum_{m\in[M]}x_{m,n}(t)a_n(t)=u_n(t)a_n(t)$$

Combining the above two equation, we have:

$$\sum_{m\in[M]}\sum_{n\in[N]}x_{m,n}(t)a_n(t)=\sum_{n\in[N]}u_n(t)a_n(t),\quad\forall t\in[T] \qquad (28)$$

Then, we consider term $\sum_{m\in[M]}\sum_{n\in[N]}\sum_{k\in\{0,1\}}x_{m,n}(t)c_{n,k}b_{m,k}(t)$. If $b_{m,k}(t)=0$, we see:

$$\sum_{n\in[N]}x_{m,n}(t)c_{n,k}b_{m,k}(t)=C_kb_{m,k}(t)$$

Authorized licensed use limited to: University of Wollongong. Downloaded on April 04,2020 at 13:46:08 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Otherwise, if $b_{m,k}(t) > 0$, by condition (10.4), we have $\sum_{n\in[N]} c_{n,k}x^*_{m,n}(t) = C_k$. Based on it, we obtain that:

$$\sum_{n\in[N]} x_{m,n}(t)c_{n,k}b_{m,k}(t) = C_k b_{m,k}(t)$$

Take the above two conditions into consideration, we arrive at the conclusion that:

$$(24) = \eta(1+\frac{\epsilon}{MN}) \times(\sum_{n\in[N]} u_n(t)a_n(t) - \sum_{m\in[M]}\sum_{k\in\{0,1\}} C_k b_{m,k}(t)) \quad (29)$$

Summing up (29) we have:

$$\sum_{t\in[T]} C_{deploy}(t) = \eta(1+\frac{\epsilon}{MN})\sum_{t\in[T]} \times(\sum_{n\in[N]} u_n(t)a_n(t) - \sum_{m\in[M]}\sum_{k\in\{0,1\}} C_k b_{m,k}(t)) \quad (30)$$

which is $\eta(1+\frac{\epsilon}{MN})$ times of D. Therefore, we show that deployment cost of P is no larger than $\ln(1+\frac{MN}{\epsilon})(1+\frac{\epsilon}{MN})$ times of D. □

APPENDIX G
PROOF OF LEMMA 3

Proof: We considered the two phases respectively. If the current iteration $h+1$ is in **Phase 1**, we have:

$$\begin{aligned}\mathbf{E}[(x^{h+1}_{m,n}(t))] &= \frac{\beta}{\beta+\alpha}(x^h_{m,n}(t)+\alpha r_{m,n}) \\ &\quad + \frac{\alpha}{\beta+\alpha}(x^h_{m,n}(t)-\beta r_{m,n}) \\ &= x^h_{m,n}(t)\end{aligned}$$

If the current iteration $h+1$ is in **Phase 2** and $x^{h+1}_{m,n}(t)$ is in the matchings, we have:

$$\begin{aligned}\mathbf{E}[(x^{h+1}_{m,n}(t))] &= \frac{\beta}{\beta+\alpha}(x^h_{m,n}(t)+\alpha) + \frac{\alpha}{\beta+\alpha}(x^h_{m,n}(t)-\beta) \\ &= x^h_{m,n}(t)\end{aligned}$$

or

$$\begin{aligned}\mathbf{E}[(x^{h+1}_{m,n}(t))] &= \frac{\beta}{\beta+\alpha}(x^h_{m,n}(t)-\alpha) + \frac{\alpha}{\beta+\alpha}(x^h_{m,n}(t)+\beta) \\ &= x^h_{m,n}(t)\end{aligned}$$

In conclusion, the expectation of each variable $x_{m,n}(t)$ remains the same throughout RA. □

APPENDIX H
PROOF OF LEMMA 4

Proof:

$$\begin{aligned}\mathbf{E}[\bar{P}(operate)] &= \mathbf{E}[\sum_{t\in[T]}\sum_{m\in[M]}\sum_{n\in[N]} O_n\bar{x}_{m,n}(t)] \\ &= \sum_{t\in[T]}\sum_{m\in[M]}\sum_{n\in[N]} O_n\mathbf{E}[\bar{x}_{m,n}(t)] \\ &= \sum_{t\in[T]}\sum_{m\in[M]}\sum_{n\in[N]} O_n x^*_{m,n}(t) \\ &\le P(ORFA)\end{aligned}$$

The second equality is due to Lemma 3. □

APPENDIX I
PROOF OF LEMMA 5

Proof:

$$\begin{aligned}\mathbf{E}[\bar{P}(deploy)] &= \mathbf{E}[\sum_{t\in[T]}\sum_{m\in[M]}\sum_{n\in[N]} D_n\bar{x}_{m,n}(t)] \\ &= \sum_{t\in[T]}\sum_{m\in[M]}\sum_{n\in[N]} D_n\mathbf{E}[\bar{x}_{m,n}(t)] \\ &= \sum_{t\in[T]}\sum_{m\in[M]}\sum_{n\in[N]} D_n x^*_{m,n}(t) \\ &\le \sum_{t\in[T]}\sum_{m\in[M]}\sum_{n\in[N]} \max_{n\in[N]}\{\frac{D_n}{O_n}\}O_n x^*_{m,n}(t) \\ &\le \sum_{t\in[T]}\sum_{m\in[M]}\sum_{n\in[N]} \Omega O_n x^*_{m,n}(t) \\ &\le \Omega P(ORFA)\end{aligned}$$

□

APPENDIX J
PROOF OF LEMMA 6

Proof: We calculate the partial derivative of objective function F_{obj} of problem (9) with respect to each variable $x_{m,n}(t)$

$$\frac{\partial F_{obj}}{\partial x_{m,n}(t)} = O_n + \frac{D_n}{\eta}\ln\frac{x_{m,n}(t)+\frac{\epsilon}{MN}}{x_{m,n}(t-1)+\frac{\epsilon}{MN}}$$

Thus, we have:

$$\begin{aligned}&\frac{\partial F_{obj}}{\partial x_{m,n}(t)} \\ &= O_n + \frac{D_n}{\eta}\ln\frac{x_{m,n}(t)+\frac{\epsilon}{MN}}{x_{m,n}(t-1)+\frac{\epsilon}{MN}} \\ &= Dn(\frac{O_n}{D_n} + \frac{1}{\eta}\ln\frac{x_{m,n}(t)+\frac{\epsilon}{MN}}{x_{m,n}(t-1)+\frac{\epsilon}{MN}}) \\ &\ge \frac{D_n}{\eta}(\ln(\frac{X_{max}MN}{\epsilon}+1) + \ln\frac{x_{m,n}(t)+\frac{\epsilon}{MN}}{x_{m,n}(t-1)+\frac{\epsilon}{MN}})\end{aligned}$$

where the inequality is because of Assumption 1.

Furthermore, we have:

$$\ln\frac{x_{m,n}(t)+\frac{\epsilon}{MN}}{x_{m,n}(t-1)+\frac{\epsilon}{MN}} \ge \ln\frac{\frac{\epsilon}{MN}}{X_{max}+\frac{\epsilon}{MN}}$$

Therefore, we observe:

$$\begin{aligned}\frac{\partial F_{obj}}{\partial x_{m,n}(t)} &\ge \frac{D_n}{\eta}(\ln(\frac{X_{max}MN}{\epsilon}+1) + \ln\frac{x_{m,n}(t)+\frac{\epsilon}{MN}}{x_{m,n}(t-1)+\frac{\epsilon}{MN}}) \\ &\ge \frac{D_n}{\eta}(\ln(\frac{X_{max}MN}{\epsilon}+1) + \ln\frac{\frac{\epsilon}{MN}}{X_{max}+\frac{\epsilon}{MN}}) \\ &= 0\end{aligned}$$

In conclusion, we see that F_{obj} is always increasing as each variable $x_{m,n}(t)$ increases. Thus, constraint (9a) is always tight, *i.e.*, constraint (9a) becomes equality constraint. If constraint (9a) is not tight, we can always remove the overflowing part,*i.e.*, $\sum_{m\in[M]} x_{m,n}(t) - u_n(t)$, to reduce objective value.

Now, if there exists such VNF n needing to be rounded is assigned fractionally to exactly one server, then $\sum_{m\in[M]} x_{m,n}(t)$ is fractional, which contradicts the above conclusion. Thus, we prove the correctness of Lemma 6. □

Authorized licensed use limited to: University of Wollongong. Downloaded on April 04,2020 at 13:46:08 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

APPENDIX K
PROOF OF LEMMA 7

Proof: We start by considering the first iteration entering **Phase 2**. At the iteration, we have $|V'| \leq |N'| + 2|M'|$. Also we observe that $|V'| \geq 2|N'|$ due to Lemma 6 and $|V'| \geq 4|M'|$ based on definition. Leveraging above observations, we have $|V'| = 2|N'| = 4|M'|$. Thus, we ensure that each key server is assigned with exactly 4 floating VNFs. Every other floating server is with less than 4 floating VNFs. In the following iterations, the number of floating VNFs assigned to each floating servers only reduces. □

APPENDIX L
PROOF OF LEMMA 8

Proof: For each floating server or floating VNF, if it only has one floating edge, it is easy to observe that Lemma 8 holds. Now, suppose the floating server or floating VNF has at least two floating edges. If the floating server or VNF is in the path/even circle, then it must has exactly two floating edges in it. And there must be one edge in Ψ_1 and the other edge in Ψ_2. Since the edge in Ψ_1 increases/decreases the same amount as the edge in Ψ_2 decreases/increases, the overall amount remains the same. Thus, we show that Lemma 8 holds. □

APPENDIX M
PROOF OF THEOREM 7

Proof: Since every iteration h in **Phase 1** respects the linear system (11) and (12), we have:

$$\sum_{m\in[M]} x^h_{m,n}(t) = \sum_{m\in[M]} x^*_{m,n}(t) = u_n, \quad \forall n \in [N], h \text{ in } \textbf{Phase 1}. \tag{31}$$

Because u_n is always an integer, we can combine equation (31) with Lemma 8 to show:

$$\sum_{m\in[M]} \bar{x}_{m,n}(t) = \sum_{m\in[M]} x^*_{m,n}(t) = u_n, \forall n \in [N]$$

□

APPENDIX N
PROOF OF THEOREM 8

Proof: During iterations in **Phase 1**, it is easy to see that our rounding method respects:

$$\sum_{n\in[N]} c_{n,k} x^h_{m,n}(t) = \sum_{n\in[N]} c_{n,k} x^*_{m,n}(t), \forall m \in [M], k \in \{0,1\}$$

Thus, we focus on **Phase 2**. Based on Lemma 7, each server is with at most four floating edges.

We first investigate floating machine m with four floating edges at the beginning iteration h of **Phase 2**. Let the four floating jobs be j_1, j_2, j_3 and j_4. And let the fractional part of $x_{m,j_1}(t)$, $x_{m,j_2}(t)$, $x_{m,j_3}(t)$ and $x_{m,j_4}(t)$ be ρ_1, ρ_2, ρ_3 and ρ_4. There are four possible cases:

▷ **Case 1:** All four fractional parts are rounded up.

Based on Lemma 8, we have:

$$(1-\rho_1) + (1-\rho_2) + (1-\rho_3) + (1-\rho_4) \leq 1 \tag{32}$$

And the additional resource k consumption due to rounding is:

$$\begin{aligned} &c_{j_1,k}(1-\rho_1) + c_{j_2,k}(1-\rho_2) + c_{j_3,k}(1-\rho_3) + c_{j_4,k}(1-\rho_4) \\ &\leq \max_{n\in[N]} c_{n,k}((1-\rho_1) + (1-\rho_2) + (1-\rho_3) + (1-\rho_4)) \\ &\leq \max_{n\in[N]} c_{n,k} \end{aligned}$$

where the last inequality is due to (32).

▷ **Case 2:** Three fractional parts are rounded up while one fractional part is rounded down.

Without loss of generality, we assume ρ_1, ρ_2 and ρ_3 are rounded up and ρ_4 is rounded down. And because of Lemma 8, we ensure:

$$(1-\rho_1) + (1-\rho_2) + (1-\rho_3) - \rho_4 \leq 1 \tag{33}$$

The additional resource k consumption is:

$$\begin{aligned} &c_{j_1,k}(1-\rho_1) + c_{j_2,k}(1-\rho_2) + c_{j_3,k}(1-\rho_3) - c_{j_4,k}\rho_4 \\ &\leq \max_{n\in[N]} c_{n,k}((1-\rho_1) + (1-\rho_2) + (1-\rho_3)) - \min_{n\in[N]} c_{n,k}\rho_4 \\ &\leq \max_{n\in[N]} c_{n,k} + (\max_{n\in[N]} c_{n,k} - \min_{n\in[N]} c_{n,k})\rho_4 \\ &< 2\max_{n\in[N]} c_{n,k} - \min_{n\in[N]} c_{n,k} \end{aligned}$$

The second inequality is based on (33), and the last inequality holds since $\rho_4 \in (0,1)$.

▷ **Case 3:** Two fractional parts are rounded up while two fractional parts are rounded down.

We assume ρ_1 and ρ_2 are rounded up, and ρ_3 and ρ_4 are rounded down. Due to Lemma 8, we see:

$$(1-\rho_1) + (1-\rho_2) - \rho_3 - \rho_4 \leq 1 \tag{34}$$

The additional resource k consumption is:

$$\begin{aligned} &c_{j_1,k}(1-\rho_1) + c_{j_2,k}(1-\rho_2) - c_{j_3,k}\rho_3 - c_{j_4,k}\rho_4 \\ &= c_{j_1,k} + c_{j_2,k} - (c_{j_1,k}\rho_1 + c_{j_2,k}\rho_2 + c_{j_3,k}\rho_3 + c_{j_4,k}\rho_4) \\ &\leq 2\max_{n\in[N]} c_{n,k} - \min_{n\in[N]} c_{n,k}(\rho_1 + \rho_2 + \rho_3 + \rho_4) \\ &\leq 2\max_{n\in[N]} c_{n,k} - \min_{n\in[N]} c_{n,k} \end{aligned}$$

The last inequality is due to (34).

▷ **Case 4:** Only one fractional part is rounded up while three fractional parts are rounded down.

Since only one fractional part is rounded up, we can easily observe that the additional resource k consumption is at most $\max_{n\in[N]} c_{n,k}$.

Therefore, additional resource k consumption for floating server with four floating edges is at most $(2\max_{n\in[N]} c_{n,k} - \min_{n\in[N]} c_{n,k})$. Floating servers with less than four floating edges can be analysed in the same way. In conclusion, we ensure that the additional resource k consumption for each floating server due to rounding steps is less than $(2\max_{n\in[N]} c_{n,k} - \min_{n\in[N]} c_{n,k})$. □

REFERENCES

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

[2] N. Buchbinder, S. Chen, and J. Naor, "Competitive analysis via regularization," in *Proc. 25th Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan. 2014, pp. 436–444.

[3] V. S. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan, "A unified approach to scheduling on unrelated parallel machines," *J. ACM*, vol. 56, no. 5, pp. 1–31, Aug. 2009.

[4] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, "Design and implementation of a consolidated middlebox architecture," in *Proc. NSDI*, 2012, p. 24.

[5] J. Martins *et al.*, "ClickOS and the art of network function virtualization," in *Proc. NSDI*, 2014, pp. 459–473.

[6] J. Hwang, K. K. Ramakrishnan, and T. Wood, "NetVM: High performance and flexible networking using virtualization on commodity platforms," in *Proc. NSDI*, 2014, pp. 445–458.

Authorized licensed use limited to: University of Wollongong. Downloaded on April 04,2020 at 13:46:08 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[7] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," in *Proc. SIGCOMM*, 2013, pp. 27–38.
[8] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The dynamic placement of virtual network functions," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, May 2014, pp. 1–9.
[9] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield, "Split/merge: System support for elastic execution in virtual middleboxes," in *Proc. NSDI*, vol. 13, 2013, pp. 227–240.
[10] A. Gember-Jacobson *et al.*, "OpenNF: Enabling innovation in network function control," in *Proc. SIGCOMM*, 2014, pp. 163–174.
[11] S. Palkar *et al.*, "E2: A framework for NFV applications," in *Proc. SOSP*, 2015, pp. 121–136.
[12] H. Moens and F. D. Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *Proc. 10th Int. Conf. Netw. Service Manage. (CNSM) Workshop*, Nov. 2014, pp. 418–423.
[13] M. Ghaznavi, N. Shahriar, S. Kamali, R. Ahmed, and R. Boutaba, "Distributed service function chaining," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2479–2489, Nov. 2017.
[14] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 1346–1354.
[15] W. Ma, O. Sandoval, J. Beltran, D. Pan, and N. Pissinou, "Traffic aware placement of interdependent NFV middleboxes," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.
[16] R. Shi *et al.*, "MDP and machine learning-based cost-optimization of dynamic resource allocation for network function virtualization," in *Proc. IEEE Int. Conf. Services Comput.*, Jun. 2015, pp. 65–73.
[17] X. Wang, C. Wu, F. Le, A. Liu, Z. Li, and F. Lau, "Online VNF scaling in datacenters," in *Proc. IEEE 9th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2016, pp. 140–147.
[18] Y. Jia, C. Wu, Z. Li, F. Le, and A. Liu, "Online scaling of NFV service chains across geo-distributed datacenters," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 699–710, Apr. 2018.
[19] X. Fei, F. Liu, H. Xu, and H. Jin, "Adaptive VNF scaling and flow routing with proactive demand prediction," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 486–494.
[20] M. Lin, A. W. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1378–1391, Oct. 2013.
[21] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[22] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan, "Dependent rounding and its applications to approximation algorithms," *J. ACM*, vol. 53, no. 3, pp. 324–360, May 2006.
[23] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *Proc. 11th Int. Conf. Netw. Service Manage. (CNSM)*, Nov. 2015.
[24] *Amazon EC2 Instance Types*. Accessed: Dec. 1, 2018. [Online]. Available: https://aws.amazon.com/ec2/instance-types
[25] *Amazon EC2 Pricing*. Accessed: Dec. 1, 2018. [Online]. Available: https://aws.amazon.com/ec2/pricing/on-demand

Ziyue Luo (Student Member, IEEE) received the B.S. degree from Wuhan University, China, in 2018. He is currently pursuing the Ph.D. degree with the Department of Computer Science, The University of Hong Kong. His research interests include cloud computing, network function virtualization, and distributed machine learning/big data analytics systems.

Chuan Wu (Senior Member, IEEE) received the B.Engr. and M.Engr. degrees from the Department of Computer Science and Technology, Tsinghua University, China, in 2000 and 2002, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Toronto, Canada, in 2008. From 2002 to 2004, she worked in the information technology industry in Singapore. Since September 2008, she has been with the Department of Computer Science, The University of Hong Kong, where she is currently an Associate Professor. Her current research is in the areas of cloud computing, distributed machine learning/big data analytics systems, network function virtualization, and intelligent elderly care technologies. She is a member of the ACM and has served as a TPC member and a reviewer for various international conferences and journals. She was a co-recipient of the best paper awards of HotPOST 2012 and ACM e-Energy 2016. She has also served as the Chair of the Interest Group on Multimedia services and applications over Emerging Networks (MEN) of the IEEE Multimedia Communication Technical Committee (MMTC) from 2012 to 2014. She is an Associate Editor of the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON MULTIMEDIA, the *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY.

Authorized licensed use limited to: University of Wollongong. Downloaded on April 04,2020 at 13:46:08 UTC from IEEE Xplore. Restrictions apply.