Traffic-aware and Energy-efficient vNF Placement for Service Chaining: Joint Sampling and Matching Approach

Chuan Pham*, Nguyen H. Tran*, Shaolei Ren†, Walid Saad*‡, Choong Seon Hong*,
*Department of Computer Science and Engineering, Kyung Hee University, Korea,
†Department of Electrical & Computer Engineering, University of California at Riverside, USA,
‡Bradley Department of Electrical and Computer Engineering, Virginia Tech, USA.

Abstract—Although network function virtualization (NFV) is a promising approach for providing elastic network functions, it faces several challenges in terms of adaptation to diverse network appliances and reduction of the capital and operational expenses of the service providers. In particular, to deploy service chains, providers must consider different objectives, such as minimizing the network latency or the operational cost, which are coupled objectives that have traditionally been addressed separately. In this paper, the problem of virtual network function (vNF) placement for service chains is studied for the purpose of energy and traffic-aware cost minimization. This problem is formulated as an optimization problem named the joint operational and network traffic cost (OPNET) problem. First, a sampling-based Markov approximation (MA) approach is proposed to solve the combinatorial NP-hard problem, OPNET. Even though the MA approach can yield a near-optimal solution, it requires a long convergence time that can hinder its practical deployment. To overcome this issue, a novel approach that combines the MA with matching theory, named as SAMA, is proposed to find an efficient solution for the original problem OPNET. Simulation results show that the proposed framework can reduce the total incurred cost by up to 19% compared to the existing non-coordinated approach.

Index Terms—Datacenters, Network Function Virtualization, Virtual Network Function, Network Traffic, Resource Allocation.

1 Introduction

Network function virtualization (NFV), an innovative network architecture paradigm, has emerged as a promising network architecture. NFV uses standard IT virtualization techniques to consolidate many network equipment types onto industry standard high volume servers, switches, and storages [1]. NFV is based on the concept of *virtual network functions* (vNFs) [1], an abstract building block whose goal is to process the network traffic to accomplish a specific task, such as a firewall or a load-balancer. Traditionally, these network functions are implemented on dedicated network devices, which are typically known as middleboxes. Although such middleboxes are able to handle heavy traffic loads, they are expensive and inflexible to implement. NFV is seen as a solution that can replace dedicated hardware platforms with software implementations in a virtualized environment. Hence, multiple and heterogeneous vNFs can be hosted on general-purpose CPUs or virtual machines (VMs) for various purposes, such as rapid service innovation, improved operational efficiencies, reducing power usage, providing standard and open interfaces, greater flexibility, and improved capital efficiencies.

Another benefit of using NFV is the simplicity in the implementation of heterogeneous network services by exploiting the important concept of *service chaining* [2], in which multiple vNFs are used in sequence to deliver a service. Each service chain (SC) includes an ordered list of vNFs (e.g., firewalls or network address translations) that are "stitched" together in the network. Based on virtualization, the software-based approach of NFV enables a much higher degree of automation, such as service deployment, on-demand resource allocation, failure detection and on-time recovery, and software upgrades [3]. vNFs and logical links between them can be easily embedded and shared on physical resources.

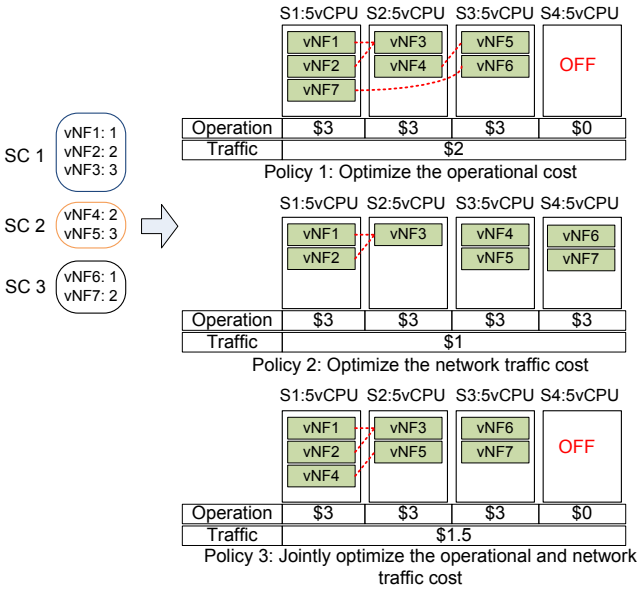


Fig. 1: Example of vNF placement with different policies.

Corresponding author: Dr. C.S. Hong (Email: cshong@khu.ac.kr).

However, an effective deployment of NFVs requires meeting several key challenges. In particular, to deploy heterogeneous network functions for SCs, the providers face several tradeoffs between different objectives, such as minimizing the latency of the network and minimizing the number of active nodes in the network. These objectives are conflicting since minimizing the number of active nodes in the consolidation policy [4] can increase the aggregation traffic on the physical links and nodes that leads to inefficiencies for the network latency objective [5]. Meanwhile, optimizing the network latency increases the cost incurred in the system due to spending more resources to deploy vNFs.

For instance in the consolidation policy, the vNF function placement is needed to reduce the *operational cost*, which includes license fees, redundant resources, or power consumption. By using the consolidation policy in middleboxes, the resource provisioning cost can be reduced from 1.8 to 2.5 times, as shown in [4]. Another efficiency implementation of consolidation policy is shown in the Oracle datacenter [6], where they can increase the chiller efficiency by 30%, remove lead and chemical waste, and reduce the free cooling per year by more than one-third. However, when formalizing a request for chaining several vNFs together, the network providers cannot ignore the possible dependencies among them. Relying only on the consolidation of resource usage may cause congestion in the physical network since minimizing the number of active nodes increases the additional used bandwidth of all embedded SCs on the physical links [7]. Moreover, when deploying vNFs on a cloud, the consolidation policy (e.g., used in OpenStack clouds [8]) also faces to the significant migration cost including migration process costs, reconfiguration systems, or network congestion overhead.

On the other hand, only a handful of solutions exists for the service providers that focus on the network traffic between vNFs, such as in [5], [7], and [9]. In particular, all network flows of SCs must be monitored and the obvious choice is to deploy vNFs on physical nodes with minimum *network traffic cost* [7]. In fact, the relationships between vNFs of SCs are complicated. Some vNFs of different SCs can be shared in implementation (e.g., anti-virus functions), while other types of vNFs cannot be shared (e.g., a firewall) [7]. Furthermore, some vNFs can modify or change the traffic between vNFs, e.g., a firewall can drop incoming packets that violate the security policies, or a video transcoding can change the packet sizes [10]. To simplify the complexity of SCs, here, we consider unshared vNFs, where a shared vNF can be represented by replicated vNFs with the same vNF instances implemented on VMs. This consideration is possible in the virtualization environment, where the shared vNF often requires a double resource on the VM [11]. Further, even though implementing vNFs on the virtualization environment of cloud datacenters facilitates vNF implementation, a design that only considers to optimize the network traffic cost, can lead to the inefficient and fragmented resource usage (e.g., CPU, memory), which can, in turn, lead to important resource allocation challenges within the datacenters [12].

We now illustrate the benefit of vNF placement in a joint consideration of both mentioned policies by showing a simple example. Fig. 1 presents three possible policies of vNF placements by deploying SCs that include lists of vNFs. For ease of exposition, we give the example with only one resource type (vCPU) of vNFs and physical nodes. Assume that the cost to operate one physical node (in terms of power) is $\$3$/hour and that network delay cost is $\$0.5$/link. Hence, the total cost to implement all SCs is as follows:

a) Policy 1: Based on consolidation and no network traffic-aware consideration, there is one idle node, while the system must serve four interconnections between vNFs. Thus, the service provider will be charged $\$11$ to host all vNFs.
b) Policy 2: Based on network traffic policy and no consolidation, the number of interconnections can be reduced, however, all nodes have to turn-on. Thus, the service provider will be charged $\$13$ as the highest total cost compared to other cases.
c) Policy 3: Under both operational cost and network traffic cost consideration, the service provider will be charged $\$10.5$ as the smallest cost to host all vNFs, since the number of interconnections and the number of active nodes are reduced.

The example shows that jointly optimizing these policies, which are traditionally studied separately, increases the efficiency of vNF placement. Moreover, the necessity of the joint operational and network traffic cost is not well-studied in the NFV literature [7], [13]. Therefore, in this paper, we devise a traffic-aware and energy-efficient vNF placement for service chaining that optimizes both the operational and network traffic cost, considering the heterogeneous physical nodes and workload. In summary, our key contributions are as follows:

- We first formulate the joint operational and network traffic cost as an optimization problem (named OPNET) whose goal is to minimize the total cost of the network.
- To derive the solution for OPNET, we first propose an algorithm based on the Markov approximation (MA) technique to solve the combinatorial vNF placement problem OPNET. However, this method is inefficient to be used directly because it is then shown to exhibit a long convergence time to find the near-optimal solution. This is due to the large state space of OPNET, which depends on the combination between chosen subsets of nodes and vNF placement schemes. To overcome this challenge, we propose a novel framework for vNF placement based on the combination of the MA technique and the matching approach, named the SAMA algorithm. In SAMA, we solve OPNET by iteratively executing two steps i) finding the subset of nodes to deploy vNFs and ii) placing vNFs to minimize the total cost incurred in the system. This approach reduces the state space of feasible solutions that directly impacts to the convergence time.
- In addition, the vNF placement in the second step of SAMA can reduce the computational cost by formulated as a many-to-one matching game, where vNFs and nodes are seen the players of the proposed matching game. To solve the vNF many-to-one

matching game, we propose two approaches, including the centralized solution and the distributed solution. The centralized solution can be implemented at the SDN controller of NFV architecture. In contrast, the distributed approach can be used in the distributed system, where active nodes are equipped monitoring and scheduling functions to handle vNF placement for themselves.

- Finally, we compare our approach with state-of-the-art methods in several case studies. Simulation results show that SAMA can effectively optimize the total cost in a specific time slot and over a long-term consideration. Moreover, the result shows that SAMA converges more quickly than the first approach using only MA. Furthermore, compared to existing techniques, SAMA is shown to be superior in terms of reducing heterogeneous instances of vNFs and node configurations.

The rest of the paper is organized as follows. Section 2 discusses about the current works, related to our proposed method. Section 3 presents the system model and problem statement. To solve the problem, we discuss the solution applying MA method in Section 4. In Section 5, we design a heuristic algorithm that combines the MA method and the matching game to solve the optimization OPNET. The matching game approach for vNF placement is represented in Section 6. We then simulate and evaluate our work in the Section 7. Finally, we present conclusion in Section 8.

2 Related Work

Recently, NFV has received significant attention as a new way to design, deploy, and manage network services. Some previous works [7], [9] and [14] considered vNF placement as extensions of virtual network embedding (VNE) problems [15]. Chained vNFs can be modeled as graphs to be embedded into a substrate network, where each connection between vNFs can be mapped to a physical link and multiple vNFs can be mapped to the same physical node. However, the vNF placement for service chaining and VNE have different goals and constraints [14].

Beginning with the middlebox concept [16], NFV architecture aims to reduce the operational cost (CapEx and OpEx [17]) by using consolidation methods [4], [18]. Especially, NFV can be implemented on the cloud/datacenter, where the consolidation policy becomes an enterprise method [6], [8]. Since computing resources, such as vCPU, memory and storage, are explicit, they are easily monitored, managed and consolidated by resource management functions in a cloud. Therefore, the consolidation policies in existing works [4], [7], [8] often ignore the interconnection between vNFs in a SC. In terms of vNF placement on cloud/datacenters, other studies [8], [19], and [20] considered the vNF placement with the goal of reducing the number of active nodes, as is typically done for VM placement.

Another important paradigm of vNF placement is network traffic-aware placement in which many studies have proposed mechanisms based on the NFV architecture or the SDN architecture to optimize the network traffic cost, such as in [9], [14], [21], and [22]. The authors in [9] and [21] focused on network-aware vNF placement, which focuses solely on the network traffic cost, while neglecting the consolidation policy. Other works such as [14] and [22] have no explicit solutions to solve the formulated NP-hard problems for vNF placement. The authors in [7] and [21] analyzed in detail the network traffic and its impact on vNF placement; however they did not clarify them in the objective function.

In most of these existing works, the coupling between different vNF placement objectives has not been addressed. A few existing works on the NFV architecture [7], [13] considered the vNF placement problem with multiple objectives, such as the network resource cost and the license cost. However, they do not provide any effective solution to optimize such coupled objectives.

	Workload and nodes
$\mathcal{M}$	A set of active nodes, indexed by $m = 1, .., M$.
$\mathcal{C}$	A set of SCs, indexed by $c = 1, ..., C$.
$\mathcal{N}$	A set of vNFs, indexed by $n = 1, ..., N$.
$\mathcal{P}$	A set of resource types, indexed by $p = 1, ..., P$.
r_n^p, r_m^p	Amount of resource type p of vNF n and node m, respectively.
$\boldsymbol{r_n}, \boldsymbol{r_m}$	Resource vector of vNF n and node m, respectively.
$a_{nn'}^c$	The traffic rate requirement between vNF n and n'.
$D_{mm'}$	The network latency between node m to node m'.
M'	The number of active nodes in previous time slot.
Q_m	The active power of node m.
$B_{mm'}$	The traffic rate capacity of the physical link mm'.
	Optimization
$\boldsymbol{X}_{nm}^c$	A binary decision for placing vNF n of SC c on physical nodes m.
$E(\mathcal{M})$	The energy cost of active node set $\mathcal{M}$.
$W(\mathcal{M})$	The wear-and-tear cost of active node set $\mathcal{M}$.
$G(\boldsymbol{X})$	The network traffic cost with allocation scheme $\boldsymbol{X}$.
$C(\mathcal{M}, \boldsymbol{X})$	The objective function of total incurred cost.
f	A feasible configuration.
$\mathcal{F}$	A set of all feasible configuration.
C_f	System objective under configuration f.
p_f	A probability of choosing configuration f.
δ	A positive constant for log-sum-exp approximation.
$q_{(f \to f')}$	Transition rate from state of configuration f to state of configuration f'.
α	The price that converts the power to a monetary term.
β	The price that converts the wear-and-tear cost to a monetary term.
σ	The weighted parameter.
	Matching theory
μ	A matching scheme that allocates vNFs to nodes.
ω_p	Weight of resource type p.
$n \succ_m n'$	Agent m prefers agent n to agent n' in matching.
$P(n)$	The preference list of vNF n.
$P(m)$	The preference list of node m.

TABLE 1: Summary of key notations.

3 System Model and Problem Formulation

In this section, we quantitatively analyze the cost model of vNF placement in NFV architecture. A summary of the used notations is found in Table 1.

3.1 Problem formulation

In the NFV architecture, the controller receives demand workloads and makes a decision for vNF allocation in each time slot [7], [13]. However, depending on network service types, the average demand workload is different during a time slot (e.g., 1 hour [23], [24]). We assume that time is slotted and we study the system for one time period. The assumption implies that non of configuration parameters (e.g., demand workloads, network topology, physical node configurations, etc.) in the system will change within one time slot, and the proposed algorithm must converge during this slot. Then, the operator can deploy vNFs on physical nodes and implement further vNF-specific configurations in the remaining of the time slot (e.g., setup network policies for the firewall instance) [25]. This assumption is used in several works, such as in [7], [9], [13], [15], and [26].

Virtual network functions and service chains. Consider a service provider having $M^{\max}$ heterogeneous physical

Service Chain	Chained vNFs	Traffic rate requirement (a)
Web Service	NAT-FW-WOC-IDPS	100 kbps
VoIP	NAT-FW-TM-FW-NAT	64 kbps
Online Gaming	NAT-FW-VOC-WOC-IDPS	50 kbps

TABLE 2: Network traffic requirements.
NAT: Network Address Translator, FW: Firewall, TM: Traffic Monitor, WOC: WAN Optimization Controller, IDPS: Intrusion Detection Prevention System, VOC: Video Optimization Controller

Instance Type	Memory	CPU	Throughput
Firewall (small)	4 GB	2 vCPU	100 Mbps
Firewall (standard)	4 GB	8 vCPU	200 Mbps
Firewall (large)	4 GB	8 vCPU	400 Mbps
IDS	4 GB	6.5 vCPU	80 Mpbs
IPSec (standard)	4 GB	4 vCPU	268 Mpbs
IPSec (large)	4 GB	8 vCPU	580 Mpbs
WAN-opt (standard)	2 GB	2 vCPU	10 Mpbs
WAN-opt (large)	2 GB	4 vCPU	50 Mpbs

TABLE 3: vNF instances.

nodes. In each time slot, this provider runs services on a subset $\mathcal{M}$ of active nodes to serve a set $\mathcal{C}$ of SCs. Each SC $c \in \mathcal{C}$ includes an ordered list of vNFs. We denote by $\mathcal{N}$ the set of vNFs in the system and each vNF $n \in \mathcal{N}$ belongs to only one SC c. Further, to implement vNFs for an SC, we denote the traffic rate requirement between vNF n and n' of SC c by $a^c_{nn'}$ to capture the interconnection in an SC. For example, to implement a small instance of the firewall in front of a WAN optimizers with vSRX [11], the traffic rate requirement between the firewall and the WAN optimizer is 100 kbps, as shown in Table 2.

Virtual network function placement in multi-resource node constraints. We next consider a set $\mathcal{P}$ of resource types, such as CPU, memory, and bandwidth. We denote r^p_n as the resource requirement of resource type $p \in \mathcal{P}$ of vNF n. Furthermore, r^p_m and r^p_c represent as the available resource type p of node m and SC c, respectively. For ease of notation, we use $\boldsymbol{r}_c$, $\boldsymbol{r}_n$ and $\boldsymbol{r}_m$ to denote a vector resource of SC c, vNF n, and node m, respectively.

Let X^c_{nm} be a binary variable that indicates whether vNF n of SC c is located on node m ($X^c_{nm} = 1$) or not ($X^c_{nm} = 0$). We consider a resource constraint that guarantees the total resource allocation on any node must be less than the available capacity of that node as follows:

$$\sum\nolimits_{c\in\mathcal{C}} \sum\nolimits_{n\in\mathcal{N}} {r_n}^p \cdot X^c_{nm} \leq r^p_m, \forall m \in \mathcal{M}, \forall p \in \mathcal{P}. \quad (1)$$

Moreover, we assume that each vNF of a given SC c can be placed on only one active node as captured by the following constraint:

$$\sum\nolimits_{c\in\mathcal{C}} \sum\nolimits_{m\in\mathcal{M}} X^c_{nm} = 1, \forall n \in \mathcal{N}. \quad (2)$$

Last but not least, when performing vNF placement, the total traffic rate of all virtual link nn' (the links between two vNFs n and n') embedded onto the physical link mm' has to be less than the capacity of the physical path mm', as follows

$$\sum_{c\in\mathcal{C}} \sum_{\substack{n,n'\in\mathcal{N} \\ n\neq n'}} a^c_{nn'} X^c_{nm} X^c_{n'm'} \leq B_{mm'}, \forall m, m' \in \mathcal{M}, m \neq m', \quad (3)$$

where $B_{mm'}$ is the traffic rate capacity of the path between the two physical nodes m and m'. When implementing vNFs on VMs of a cloud, the traffic rate capacity $B_{mm'}$ can be measured based on the network topology in the cloud (e.g., VL2 [27]), where $B_{mm'}$ is the aggregation capacity traffic of all paths from node m to node m'.

These constraints (1), (2) and (3) are traditional constraints in vNF placement problem. Based on these constraints, we next provide suitable models for the operational and network traffic costs.

Operational cost. As discussed in [26], turning on all physical nodes can have negative effects on reducing the carbon footprint as well as the electric cost in the system. According to [28], idle servers in data centers may draw up to 60% of peak power. Therefore, controlling the number of active nodes in the system is an important challenge for vNF placement. We formulate the operational cost needed to serve user demands as the amount of *power consumption* of all active nodes, using a linear function $\alpha \sum_{m\in\mathcal{M}} Q_m$, where α is the price that converts the power to a monetary term, and Q_m is the active power of node m.

However, the controller should not frequently turn-on/off arbitrary nodes since turning nodes into sleep/off mode and bringing them back to normal operation can lead to *wear-and-tear cost*, as shown in [26], [29], [30], and [31], which is detrimental to the lifetime of physical devices. Hence, we define a cost function that linearly depends on the number of nodes that are turned-on/off and a monetary term β as follows: $\beta|M - M'|$, where M is the number of active nodes in current time slot that corresponds to the subset of active node, M' is the number of active nodes in previous time slot, and β is an average monetary weight (i.e., \$/nodes).

The wear-and-tear cost function is often not considered in the traditional consolidation problem of datacenters [8], [20], [32], [33] as well as in NFV architecture [4], [18], where the authors usually optimize the number of active servers or physical nodes and ignore this cost incurred in the system. The tradeoff between reducing the amount of active nodes and mitigating the wear-and-tear cost must be concretely accounted for during vNF placement.

Therefore, we formulate the operational cost in the system as the aggregation cost as follows:

$$E(\mathcal{M}) = \beta|M - M'| + \sum\nolimits_{m\in\mathcal{M}} \alpha Q_m. \quad (4)$$

Traffic cost. In a service chain, an vNF needs to forward packets to another vNF depending on the virtual connection between them. These virtual connections are embedded on the active physical links of nodes. Each active physical link can have a different hop distance depending on the network topology in a cloud [27]. Hence, in this work, we formulate the network traffic cost, which is calculated via the cost to operate each active physical link based on the hop distance and the total allocated bandwidth of virtual links embedded on that physical link. Given the internal traffic that circulates between vNFs, we measure the network traffic cost based on the hop distance $D_{mm'}$ between the physical nodes m and m'. The hop distance of the network topology in a cloud is well-studied in [27], [34]. Hence, the network traffic cost function can be written as follows:

$$G(\boldsymbol{X}) = \sum_{\substack{m,m'\in\mathcal{M} \\ m\neq m'}} D_{mm'} \sum_{c\in\mathcal{C}} \sum_{\substack{n,n'\in\mathcal{N} \\ n\neq n'}} a^c_{nn'} X^c_{nm} X^c_{n'm'}, \quad (5)$$

where $\boldsymbol{X}$ is the matrix allocation.

The traffic cost is an important objective function in vNF placement. In fact, given thousands of nodes and vNFs in practice, an inefficient vNF placement will lead to high overhead of inter-traffic among vNFs. However, minimizing the used network bandwidth increases the number of physical nodes to be deployed, which has negative effects on the objective of the consolidation policy [7]. Therefore, a dynamic vNF placement, controlling both operational and network traffic costs, is a significant issue in NFV architecture.

We now combine the two cost models above using a weight factor $\sigma \in [0,1]$ in a system-wide objective function $C(\mathcal{M}, \boldsymbol{X})$ as follows:

$$C(\mathcal{M}, \boldsymbol{X}) = \sigma E(\mathcal{M}) + (1-\sigma) G(\boldsymbol{X}). \tag{6}$$

The design parameter σ can be adjusted to achieve any desired performance/cost tradeoff. For example, a larger σ will allow the system to emphasize more the optimization of the operational cost, while a smaller σ stresses network traffic cost minimization.

Based on the vNF placement constraints and cost models, we formulate the joint operational and network traffic problem (OPNET) as follows:

$$\begin{aligned}
\text{OPNET}: \min_{\mathcal{M},\boldsymbol{X}} \quad & C(\mathcal{M}, \boldsymbol{X}), \\
\text{s.t.} \quad & \sum_{c\in\mathcal{C}} \sum_{n\in\mathcal{N}} {r_n}^p \cdot X_{nm}^c \le r_m^p, \forall m \in \mathcal{M}, \forall p \in \mathcal{P}, \\
& \sum\nolimits_{c\in\mathcal{C}} \sum\nolimits_{m\in\mathcal{M}} X_{nm}^c = 1, \forall n \in \mathcal{N}, \\
& \sum_{c\in\mathcal{C}} \sum_{\substack{n,n'\in\mathcal{N} \\ n\neq n'}} a_{nn'}^c X_{nm}^c X_{n'm'}^c \le B_{mm'}, \\
& \forall m, m' \in \mathcal{M}, m \neq m', \\
& X_{nm}^c = \{0,1\}, \forall c \in \mathcal{C}, \forall n \in \mathcal{N}, \forall m \in \mathcal{M}, \\
& |\mathcal{M}| \le M^{\max}.
\end{aligned} \tag{7}$$

OPNET aims to find a vNF placement to minimize the total cost incurred in the system, which is represented in the objective function as the summation of the operational and traffic cost. However, the problem above cannot be found in the polynomial time since it is NP-hard. Among the many choices of optimization methods, the one we advocate below, the MA-based approach has an advantage in giving an approximated solution for the high complexity of the combinatorial optimization problem.

4 Markov Approximation Algorithm for OPNET

4.1 Log-sum-exp Approximation

Let $f = \{\mathcal{M}, \boldsymbol{X}\}$ be a configuration for the joint vNF consolidation and network traffic-aware placement, and $\mathcal{F}$ be the set of feasible configuration defined by constraints (1), (2) and (3). Configuration f indicates a specific vNF mapping scheme on a subset of active nodes $\mathcal{M}$. A change of any vNF in the allocation scheme will create a new configuration (or new state in the Markov chain). We show a simple example of Markov chain with three vNFs and two nodes in Fig. 2. To deploy two SCs, such as SC1:(vNF 1, vNF 2) and SC2:(NF3), Fig. 2 depicts the transition states between feasible configurations. For ease of presentation, we let $C_f = C(\mathcal{M}, \boldsymbol{X})$. Thus, we have $\min_{f\in\mathcal{F}} C_f$, which can be rewritten as follows

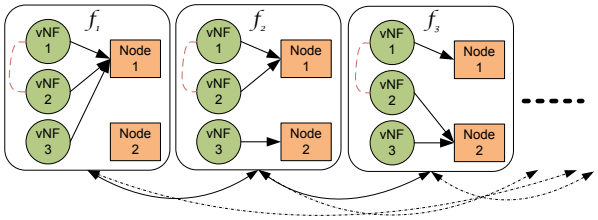


Fig. 2: A simple SC scenario with three vNFs and two physical nodes. Each state is represented by a specific allocation of three vNFs onto 2 nodes. From a given configuration f_1 with three vNFs placed on node 1, the transition from f_1 to f_2 is represented by one changed assignment (vNF 3 changes from node 1 to node 2). Similarly, the transition from one configuration to another is represented by one vNF that has changed its allocation.

$$\min_{p>0} \sum\nolimits_{f\in\mathcal{F}} p_f C_f, \tag{8a}$$

$$\text{s.t.} \sum\nolimits_{f\in\mathcal{F}} p_f = 1, \tag{8b}$$

where p_f is the probability of choosing configuration f (i.e., its weight). Following the framework, we apply log-sum-exponential approximation [35] to the OPNET problem as follows:

$$\begin{aligned}
\min_{p>0} & \sum\nolimits_{f\in\mathcal{F}} p_f C_f + \frac{1}{\delta} \sum\nolimits_{f\in\mathcal{F}} p_f \log(p_f), \\
\text{s.t.} & \sum\nolimits_{f\in\mathcal{F}} p_f = 1,
\end{aligned} \tag{9}$$

where δ is a large positive constant.

Based on the analysis about the optimization problem (9) in [35], we obtain the optimal probability distributions p^* as follows

$$p_f^*(C_f) = \frac{\exp(-\delta C_f)}{\sum_{f'\in\mathcal{F}} \exp(-\delta C_{f'})}, \forall f \in \mathcal{F}, \tag{10}$$

and the optimal objective value is

$$-\frac{1}{\delta} \log \left[\sum_{f\in\mathcal{F}} \exp(-\delta C_f) \right] \approx \min_{f\in\mathcal{F}} C_f. \tag{11}$$

Unfortunately, to calculate (10), it requires complete information on $\mathcal{F}$, which is difficult to find in practice due to the large solution space. Thus, we view f as a state of a time reversible of Markov chain. The key idea to create an algorithm is to treat $p_f^*(C_f)$ as the stationary distribution of a time reversible Markov chain. As the Markov chain converges to its stationary distribution, we approach $p_f^*(C_f)$ as an optimal solution.

4.2 Markov approximation as a solution for the OPNET problem

Markov chain and transition rate. The next step in the Markov approximation framework is to design a problem specific Markov chain, in which states and transmission

rates between states are defined. Each state (configuration) f in the Markov chain represents a feasible configuration with its corresponding stationary distribution $p_f^*(C_f)$ given in (10), and a set $\mathcal{F}$ of states represents all feasible configurations. Since the configurations will be time-shared according to p_f^* when the Markov chain converges, the controller will operate in the best configurations most of the time [35]. Therefore, based on (10), the configurations with low cost will have high probability, and thus, the operator will use those configurations more often. It was proven in [35] that for any probability distribution of the product form $p_f(C_f)$ given in (10), there exists at least one continuous-time time-reversible ergodic Markov chain whose stationary distribution is $p_f(C_f)$.

Consider two configurations $f, f' \in \mathcal{F}$ that represent the states of the time-reversible ergodic Markov chain with stationary distribution $p_f^*(C_f)$, we then derive the transition probability between these states as follows. We define $q_{(f\to f')}$ and $q_{(f'\to f)}$ the non-negative transition rates from $f \to f'$ and $f' \to f$, respectively, as shown in Fig. 3. Then, the particular form of (8b) allows us to restrict our design to a time-reversible Markov chain [35], which must satisfy the following balance equations for all $f, f' \in \mathcal{F}$:

$$\begin{aligned} p_f^*(C_f) q_{(f\to f')} &= p_{f'}^*(C_f') q_{f'\to f}, \\ \exp(-\delta C_f) q_{(f\to f')} &= \exp(-\delta C_{f'}) q_{(f'\to f)}. \end{aligned} \tag{12}$$

Let C_f and $C_{f'}$ be the total cost of two state configurations f and f', respectively. Based on (12), we have

$$\begin{aligned} q_{(f\to f')} &= \exp(-\tau) \cdot \frac{1}{1+\exp\left[-\delta(C_f - C_{f'})\right]}, \\ q_{(f'\to f)} &= \exp(-\tau) \cdot \frac{1}{1+\exp\left[-\delta(C_{f'} - C_f)\right]}, \end{aligned} \tag{13}$$

where τ is a constant [35].

Therefore, we have three cases of transition as follows. First, if $C_f > C_{f'}$, then the controller chooses the new configuration f' with $q_{(f\to f')} \approx 1$. Second, if $C_f < C_{f'}$, then the controller keeps the current configuration f with $q_{(f'\to f)} \approx 1$. Finally, if $C_f = C_{f'}$, then the controller chooses the current configuration f or the new configuration f' with equal probability.

In particular, the key design challenge for constructing such a Markov chain is to ensure (i) any two states are reachable from one to another; and (ii) the balance equation (12) is satisfied. Therefore, in the implementation, we only allow links connecting two states that can be reached by performing only one change of vNF in the allocation scheme, as shown in Fig. 2. Next, we describe the proposed algorithm based on the MA method.

Markov approximation-based algorithm. The algorithm starts with an arbitrarily chosen feasible assignment solution f, and may move to another feasible solution f' according to the transmission rate $q_{(f\to f')}$. The convergence of this mechanism occurs when the Markov chain reaches to the steady-state distribution $p_f^*(C_f)$. The mechanism proceeds as follows.

Step 1: Initialize a feasible configuration f_0 with the subset $\mathcal{M}_0$ and the assignment $\boldsymbol{X}_0$. Set f_0, $\boldsymbol{X}_0$, and $\mathcal{M}_0$ to the best configuration f^*, the best vNF assignment and the best subset of active nodes, respectively.

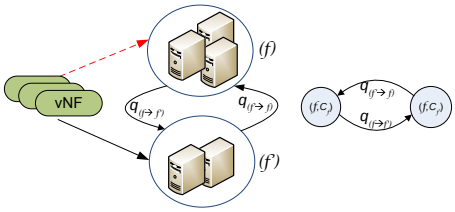


Fig. 3: Transition rates from the configuration f to f'.

Step 2: Randomly pick a vNF n, then migrate vNF n to a new configuration f', corresponding to a new active set $\mathcal{M}'$ and a new assignment $\boldsymbol{X}'$.

Step 3: Calculate $C_{f'}$ and probabilistically choose the new configuration f' according to (13). Set $f', \mathcal{M}'$ and $\boldsymbol{X}'$ to the best configuration f^* and the best subset of active nodes $\mathcal{M}^*$ and the best vNF assignment $\boldsymbol{X}^*$, respectively. Otherwise, the current configuration f^* is kept to the next iteration.

Step 4: Return to **Step 2** until the stopping criteria are met.

The MA algorithm solves OPNET in a repeated manner with two phases. The first phase is to find a random feasible configuration to implement. Then, the second phase is to compare the current cost with the previously achieved cost. The controller chooses a configuration, which is owning the smaller cost base on (13). These phases are repeated until the underlying Markov chain converges to the stationary distribution.

4.3 Discussions

Even though the MA framework can find a near-optimal solution of the NP-hard combinatorial optimization problem as proven in [35], it can exhibit a very slow convergence due to the exploration of a large number of feasible configurations. Furthermore, the multi-resource node constraints and the heterogeneous physical nodes and workload consideration further escalates the number of states in the Markov chain that draws the convergence of the algorithm becoming more decelerated.

As shown in (9), δ is to used to manage the tradeoff between exploring solutions and exploiting the current solution. When δ increases, a better solution is kept with a higher probability, leading to a more aggressive scheme. However, this leads to a high cost of taking more iterations to explore the solution space $\mathcal{F}$. Since the system stays on more exploitation and is stuck in a local optimum for a long time before successfully exploring other better solutions.

Given a value of δ, we are implicitly solving an approximated version of the OPNET problem with an entropy term $\frac{1}{\delta}\sum_{f\in\mathcal{F}} p_f \log(p_f)$. The optimality gap is thus bounded by $\frac{1}{\delta}\log|\mathcal{F}|$, where $|F|$ is the size of the configuration set $\mathcal{F}$. It emphasizes that the optimality gap will be linearly increasing when $|\mathcal{F}|$ is exponential increasing with input size for a given δ. Consider a specific case with maximum K subsets of active nodes such that each subset has enough resources to embed N vNFs. We illustrate the optimality gap as follows. For each subset of active nodes, there exists $N!$ permutations of vNF assignments corresponding to the configuration states f. Hence, with a large subset K, the size of the state space now is $N!K$. This number of states significantly affects to the optimality gap. In the

worst case with $M^{\max}$ nodes in the system, the state space becomes $N!2^{M^{\max}}$ then the gap is $\frac{1}{\delta}\log N!2^{M^{\max}}$, which is $O(N\log N + M^{\max})/\delta$.

This issue can be solved by choosing a large δ to reduce the optimality gap [35]. However, the overhead of the long convergence by using a large δ should be carefully considered. Therefore, we next apply the MA method in a different direction, combining it with the matching approach to reduce the search space and computational complexity.

5 Joint Markov Approximation and Matching Approach for OPNET

To address the slow convergence challenge of the MA solution, we next introduce a new way to apply MA that can reduce the state space, named SAMA.

OPNET couples two variables $\mathcal{M}$ and $\boldsymbol{X}$ which, in turn, results in a large feasible state space as mentioned before. Based on a sampling-based Markov approximation framework, we find the optimal subset $\mathcal{M}$ of nodes. Simultaneously, when transitioning from state f to f', the controller must find an assignment $\boldsymbol{X}$ corresponding to the chosen subset $\mathcal{M}$ such that the total cost is minimized. Next, the procedure of SAMA is presented in details.

- *Initialize.* The controller randomly chooses a subset $\mathcal{M}_0$ of active nodes that has enough resources to deploy vNFs. Then, set $\mathcal{M}^* \leftarrow \mathcal{M}_0$.
- *Calculate the current state*. Based on the chosen subset $\mathcal{M}^*$, the minimum cost C_f^* is obtained corresponding to the configuration f^* by solving the OPNET problem.
- *Calculate the new state*. To move on the next state f', the controller randomly chooses the new subset $\mathcal{M}'$, satisfying the resource demands. Next, OPNET is solved to obtain the new assignment $\boldsymbol{X}'$ and the optimal value $C_{f'}$. Given the chosen subset $\mathcal{M}'$, OPNET can be transformed into subproblem **P1** as follows.

$$\textbf{P1:} \min_{\boldsymbol{X}} \quad C(\mathcal{M}', \boldsymbol{X}) \tag{14}$$

$$\text{s.t.} \quad \sum_{c\in\mathcal{C}}\sum_{n\in\mathcal{N}} r_n^p \cdot X_{nm}^c \le {r_m}^p, \forall c \in \mathcal{C},$$

$$\forall m \in \mathcal{M}', \forall p \in \mathcal{P}, \tag{15}$$

$$\sum_{c\in\mathcal{C}}\sum_{m\in\mathcal{M}'} X_{nm}^c = 1, \forall n \in \mathcal{N}, \tag{16}$$

$$\sum_{c\in\mathcal{C}}\sum_{\substack{n,n'\in\mathcal{N}\\ n\neq n'}} a_{nn'}^c X_{nm}^c X_{n'm'}^c \le B_{mm'},$$

$$\forall m, m' \in \mathcal{M}', m \neq m', \tag{17}$$

$$X_{nm}^c = \{0,1\}, \forall n \in \mathcal{N}, \forall m \in \mathcal{M}'. \tag{18}$$

Subproblem **P1** is still NP-hard. Hence, we will use the framework of matching theory [36] to solve subproblem **P1** in polynomial time, where **P1** can be cast as a the generalized assignment problem [37], as formulated in Section 6.

- *Choosing the new configuration* f'. Based on the optimal value $C_{f'}$, the controller probabilistically chooses the new configuration f' according to the transition probability (13). Set f', $\mathcal{M}'$, and $\boldsymbol{X}'$ to the best configuration f^* and the best subset of active nodes $\mathcal{M}^*$ and the best vNF embedding $\boldsymbol{X}^*$, respectively. Otherwise, the current configuration f^* is kept to the next iteration.
- *Convergence*. Similar to the algorithm in Section 4.2, SAMA also solves OPNET in a repeated manner with two phases: choosing randomly the subset of active nodes and assigning vNFs into nodes. Based on Step 3 of SAMA, the controller chooses a subset, which obtains the minimum cost. The algorithm will stop until reaching the convergence condition (the underlying Markov chain converges to the stationary distribution).
  Furthermore, given a set $\mathcal{M}$, an optimal vNF assignment $\boldsymbol{X}$ can be obtained by solving subproblem **P1**. Here, the state space of SAMA is now reduced from $N!2^{M^{\max}}$ to $2^{M^{\max}}$ in the worst case, which corresponds optimality gap reduces to $O(M^{\max})/\delta$.

Algorithm 1: SAMA- A joint Sampling-based Markov approximation and Matching-theoretic approach.

1. Initialization: Choose randomly a subset $\mathcal{M}_0$ of active nodes. Then, solve **P1** to obtain the optimal assignment $\boldsymbol{X}_0$ and the corresponding cost C_{f_0}. Set $f^* \leftarrow f_0$ and $C_{f*} \leftarrow C_{f_0}$.
2. Randomly choose a new subset of active nodes $\mathcal{M}'$. Obtain $\boldsymbol{X}'$ and the corresponding optimal value $C_{f'}$ by solving **P1**.
3. Compute the transition probability q according to (13).
4. Based on (13), the controller sets $\mathcal{M}^* \leftarrow \mathcal{M}'$, $\boldsymbol{X}^* \leftarrow \boldsymbol{X}'$, and $C^* \leftarrow C'$ or keeps the current state.
5. Return to Step 2 until the stopping criteria is met.

The advantage of SAMA is to reduce the number of feasible configurations f. For each subset of active nodes in the algorithm of Section 4.2, we have to transit between the permutation of vNFs to find an optimal allocation. Meanwhile, SAMA skips that step by solving directly subproblem **P1**.

6 vNF Placement as A Many-to-one Matching Game

For a chosen subset $\mathcal{M}$ of active nodes (in step 2 of SAMA) that is used to deploy a set $\mathcal{C}$ of SCs, the controller has to solve **P1**. Subproblem **P1** is similar to the bin-packing problem [38] for which an effective solution can be derived using the framework of matching theory [36], [37]. In particular, one vNF is only embedded in one node, while one node can host many vNFs depending its available configuration. Based on matching theory [39], vNFs and nodes can be seen as the two sides of players in a *many-to-one matching* game in the subproblem **P1**. This matching game has been applied successfully in another area, VM placement in datacenters, to find the stable allocation, such as in [40] and [41]. Therefore, we advocate the matching game approach to find the assignment solution for subproblem **P1**.

6.1 Matching concepts

The assignment of vNFs to nodes can be considered as an outcome of a many-to-one matching.

Definition 1. The outcome of a vNF placement in problem **P1** is a matching μ. Formally, a *matching* is a function $\mu : \mathcal{N} \cup \mathcal{M} \rightarrow 2^{\mathcal{N} \cup \mathcal{M}}$ satisfying:

- $\mu(m) \subseteq \mathcal{N}$ such that $|\mu(m)|^p \leq r_m^p, \forall m \in \mathcal{M}, \forall p \in \mathcal{P}$, where $|\mu(m)|^p$ is the amount of aggregation resources of type p of all vNFs that are matched to m.
- $\mu(n) \subseteq \mathcal{M}$ such that $|\mu(n)|^p = r_m^p$, or $|\mu(n)|^p = 0$, $\forall n \in \mathcal{N}, m \in \mathcal{M}, \forall p \in \mathcal{P}$, where $|\mu(n)|^p$ is the amount of resources of type p of node m that is matched to n ($|\mu(n)|^p = 0$ means that vNF n is unassigned).
- $n \in \mu(m)$ if and only if $\mu(n) = m, \forall n \in \mathcal{N}, m \in \mathcal{M}$.

The definition states that a matching is a many-to-one relation in the sense that each node is matched to a subset of vNFs. The objective of any matching problem is to find a *stable and efficient* matching. In this game, each player needs to specify its preferences over the opposite set depending on its goal in the network.

Service chain's preference list. Intuitively, vNFs prefer to be located in the node that has the most amount of available resources. Also, all vNFs in an SC should be deployed on the same node to reduce the inter-communication between them [13]. Hence, we consider that all vNFs in an SC have the same preference list, called the SC's preference list. Each SC $c \in \mathcal{C}$ has a complete, strict, transitive and preference relation $P(c)$ over the node set $\mathcal{M}$. Here, $m \succ_c m'$ means that SC c prefers node m to node m' and if c prefers to remain unmatched instead of being matched to node m, i.e., $\emptyset \succ_c m$, then m is said to be unacceptable to c. One SC includes a list of vNFs that have the same preference as its SC because all vNFs in such an SC want to be allocated together in the same node.

Node's preference list. Similarly, each node has a preference list over vNFs. This preference list is based on the consolidation policy in which each node prefers to improve the resource utilization by deploying more vNFs. It also implies that the node prefers assignment to unassignment. We assume that each node $m \in \mathcal{M}$ has a complete, strict, transitive preference relation $P(m)$ over the vNF set $\mathcal{N}$. Further, $n \succ_m n'$ means that node m prefers vNF n to vNF n', and if m prefers to remain unmatched instead of being matched to vNF n, i.e., $\emptyset \succ_m n$, then n is said to be unacceptable to m.

Based on [39], in order to address the matching game, we need to find a stable solution, where there does not exist any player that is not matched to another, but they prefer to be partners. In our model, the stable matching implies that there is no vNF that wants to change its current placement due to increasing the network traffic cost, while nodes also do not want to change their assignment due to wasting more computing resources.

Definition 2. A matching μ is blocked by a pair of agents (n, m) if there exists a pair (n, m) with $n \notin \mu(m)$ and $m \notin \mu(n)$ such that $n \succ_m \mu(m)$ and $m \succ_n \mu(n)$. Such a pair is called a *blocking pair* in general.

Algorithm 2: Creating a SC's preference list

Input: A set of active nodes $\mathcal{M}$, the first vertex m_0, and matrix D.
Output: The list of nodes L.

1. Start on the current node m_0, then mark m_0 as visited: $L \leftarrow \{m_0\}$.
2. Find out the shortest bath connecting current node and unvisited node $\omega \in \mathcal{M}$ based on D.
3. Set current node to ω.
4. Mark ω as visited: $L \cup \{\omega\}$.
5. Terminate if all nodes are visited.
6. Go to Step 2.
7. Return L.

Definition 3. An SC $c \in \mathcal{C}$, is saturated if all vNFs in SC c are assigned. Similarly, a node is saturated if all its capacity is utilized. If node m has available resources, then it will accept any vNF n that $r_m^p \geq r_n^p, \forall p \in \mathcal{P}$.

To satisfy the requirement from users, all SCs have to be embedded into nodes. Therefore, all stable assignments must not contain any unsaturated SC.

Definition 4. A matching is said to be *stable* if (i) there is no blocking pair and (ii) all vNFs are embedded to nodes (or all SCs are saturated).

The next step is to define the preference lists for the players.

6.2 Algorithms for creating preference lists

Service chain's preference list. Based on the definition of preference list, each SC c aims to find nodes that has enough available resources and minimum the traffic cost. Hence, we first find a node that has the best fit resources to the demand of a SC c based on the norm-based metric applied in VM placement with multiple resources [19]. The norm-based metric is calculated as follows

$$L_2 = \sum_{p \in \mathcal{P}} \omega_p (r_c^p - r_m^p)^2, \forall c \in \mathcal{C}, \forall m \in \mathcal{M}. \quad (19)$$

The intuition of L_2 can be seen as the resource deviation between node m and SC c. The smaller the value of L_2 is, the more fit the node is for being chosen to deploy c.

Next, we find the subset of nodes with minimum network traffic based on the nearest neighbors algorithm [38] from the first node. Based on the network matrix D, we build SC's preference list for each SC c as Algorithm 2, where a node has higher ranking if it has smaller distance to the previous node.

We consider a graph whose vertices are nodes and let $\boldsymbol{D}$ be the matrix that captures the weights of the graph's edges. Starting from a chosen vertex (owning the minimum L_2), we mark it as the current vertex. The next vertex is chosen and added to the preference list of SC c if it is the closest vertex from the current vertex. Repeatedly, we can find the nearest neighbors that can embed all vNFs.

Node's preference list. The physical node interest is to improve its computing resource utilization. It means that

Algorithm 3: Creating a node's preference list

Input:All vNFs in $\mathcal{N}$.
Output: $P(m), \forall m \in \mathcal{M}$.
for each node in $\mathcal{M}$ **do**
1. Find the next vNF n such that $r_m^p \geq r_n^p, \forall p \in \mathcal{P}$ and n sheds the smallest resources;
2. Add n into the preference list of node m.

end

nodes prefer assigning more vNFs to unassignment. This goal can be seen as the objective of the knap-sack problem. Therefore, we apply the Best Fit algorithm [38] to build the preference list, the well-known heuristic algorithm to solve the knap-sack problem. In particular, we find the next vNF that sheds the smallest space left when being placed into node $m \in \mathcal{M}$. The smaller the space left by the vNF, the higher the ranking it has in the preference list. The procedure for creating the node's preference list is presented in Algorithm 3.

Algorithm 4: MDM: Multi-dimension matching algorithm for subproblem **P1**

Input: $\mathcal{N}, \mathcal{M}$.
Output: Embed all vNFs in $\mathcal{N}$ to nodes in $\mathcal{M}$.
while $\exists\ c \in \mathcal{C}$, who is not saturated **do**
  while $\exists\ n \in \mathcal{N}$ is unassigned **do**
    $m \leftarrow$ Get the highest rank in $P(n)$;
    if $r_m^p \geq r_n^p, \forall p \in \mathcal{P}$ **then**
      Allocate n to m;
      $r_m^p = r_m^p - r_n^p, \forall p \in \mathcal{P}$;
    end
    else
      Find all n' to satify $n \succ_m n'$;
      Reject all n' and set n' as unassigned, also update the resource of node m:
      $r_m^p = r_m^p + r_n^{p\prime}, \forall p \in \mathcal{P}$;
      Remove n' out of $P(m)$;
      Remove m out of $P(n')$;
    end
  end
end

6.3 Many-to-one matching game for subproblem P1

After formulating the vNF placement problem as a many-to-one matching game, we propose an extension of the deferred acceptance algorithm [39] to deal with multi-dimensional resources. When matching vNFs to nodes, each node has to compare all its available resources with vNF's resources. This means that before matching node m to vNF n, the resource constraints $r_m^p \geq r_n^p, \forall p \in \mathcal{P}$, must be satisfied.

Based on the Gale-Shapley many-to-one matching algorithm [39], we design a multi-dimension matching (called MDM) algorithm to find a stable matching. Since Gale-Shapley-based algorithm can achieve stable and optimal matching for the proposed side, our algorithm can optimize the network traffic cost (the cost of **P1**) that is used to build the preference list for vNFs. First, all vNFs in each SC propose to nodes following their shared preference lists. Physical nodes then accept vNFs based on their preference lists and reject vNFs if their resources exceed the quotas. Iteratively, all unassigned vNFs propose to the next nodes based on their preference lists. One node can reject the accepted vNFs, then accepts new vNFs if the new vNFs have higher ranks in the node's preference list. Similar to the college admission algorithm [39], when the node rejects one vNF, all the other accepted vNFs with lower ranks are also rejected.

A pseudo-code implementation of our model is shown in the Algorithm 4. Inspired by the deferred acceptance algorithm [39], Algorithm 4 initializes from the unsaturated SC in the list SC that has some unassigned vNFs n (line 3). vNF n picks the most interesting node m in its preference list $P(n)$ to propose (line 4). If node m has enough resources to deploy vNF n, it will accept vNF n (lines 6-8). Otherwise, node m rejects n. Before rejecting n, node m rejects all matched vNFs n' such that $n \succ_m n'$ (lines 10-16). Then, vNF n also removes m out of its preference list $P(n)$ and restarts the proposing process. Even though MDM is a multi-dimension matching algorithm, it follows the basic principles of the deferred acceptance algorithm in [39]. Therefore, MDM can eventually converge to the stable allocation in a finite amount of steps.

6.4 Complexity analysis

We now give a brief complexity analysis for the matching MDM algorithm. As discussed in Section 6, all vNFs in SC c have the same preferences over active nodes. In each iteration, there is at least one SC that is saturated, hence MDM will terminate at C iterations. The complexity of the matching approach only depends on the algorithms creating preference lists. For SC's preference lists, the sorting step of each SC based on the nearest neighbor algorithm requires $O(M\log_2 M)$ complexity [38]. Therefore, the complexity to create the SC's preference lists is $O(CM\log_2 M)$ for the set $\mathcal{C}$ of SCs. Similarly, for a node's preference list, the complexity of Algorithm 3 is $O(MN\log_2 N)$. Consequently, the time complexity of MDM is $O(CM(C\log_2 M + N\log_2 N))$.

The MDM mechanism is designed based on the traditional matching approach that is executed in a centralized way. To reduce the overhead of the centralized controller, we then propose a distributed algorithm that enables a practical implementation that can be executed on each physical node, where resource monitoring and scheduling functions are equipped [8].

6.5 Distributed implementation for the matching game

At each iteration of the SAMA, the matching approach can reduce the computation cost of solving the subproblem **P1**. However, a distributed implementation is still necessary and highly desirable in practice, where each node can execute its calculation for selecting appropriate vNFs from the workload. Inspired by distributed stable matching algorithm in [42], our mechanism includes two procedures, the *vNF procedure* and *the node procedure*, as presented in Algorithms 5 and 6, respectively. The execution is asynchronous on each vNF n and node m.

Algorithm 5: Matching procedure of vNF n

Input: The subset $\mathcal{M}$ of active node.
Output: Match to accepted node in $\mathcal{M}$.
end $\leftarrow$ false;
while *!end and* $P(n) \neq \emptyset$ **do**
  $m \leftarrow$ Get the first node in $P(n)$
  Send a proposed message to m;
  $msg \leftarrow$ Receive a message from a node;
  if msg=*"reject"* **then**
    Delete nodes out of $P(n)$ who sends the rejected message;
  end
  if msg=*"stop"* **then**
    end $\leftarrow$ true;
  end
end

Algorithm 6: Matching procedure of node m

Input: The set $\mathcal{N}$ of vNFs.
Output: Mapping vNFs in $\mathcal{N}$ to a given node m.
Set *end* $\leftarrow$ false and $list \leftarrow \emptyset$;
while *!end* and $P(n) \neq \emptyset$ **do**
  $msg \leftarrow$ Receive messages from vNFs;
  if msg=*"propose"* **then**
    $n \leftarrow$ Get the proposed vNF index;
    if $r_n^p \leq r_m^p, \forall p \in \mathcal{P}$ and $n \in P(m)$ **then**
      Sends an accepted message to n;
      Add vNF n to $list$ and update resources of m: $r_m^p = r_m^p - r_{n'}^p, \forall p \in \mathcal{P}$;
    end
    else if $r_n^p \leq r_m^p - |list|^p$ and $n \in P(m)$ **then**
      foreach n' with $n \succ_m n'$ **do**
        Sends a rejected message to n';
        Update resources of m: $r_m^p = r_m^p + r_{n'}^p, \forall p \in \mathcal{P}$;
      end
      if $r_n^p \leq r_m^p, \forall p \in \mathcal{P}$ **then**
        Sends an accepted message to n;
        Add vNF n to $list$ and update resources of m: $r_m^p = r_m^p - r_{n'}^p, \forall p \in \mathcal{P}$;
      end
    end
    else
      Sends a rejected message to vNF n;
    end
  end
  if msg=*"stop"* **then**
    end $\leftarrow$ true;
  end
end

The vNF procedure. As shown in Algorithm 5, the procedure is performed for every vNF agent n that proposes to physical nodes based on its SC's preference list $P(n)$.

1) Propose to the first choice in its preference list.
2) Wait for the response message. If the response message is "accept", it does nothing. If the reply is "reject", it removes the sender of that message from its preference list (This means that vNF n will not propose again to the node that already rejected its proposal).
3) Return to Step 1 until receiving a stop message or the preference list be empty.

The node procedure. Similar to the vNF procedure, the node procedure is also executed at each node, but it is more complicated. The details of the node m procedure are shown in Algorithm 6 with steps as follows.

1) Wait for proposed messages from vNFs.
2) Accept vNF n that sends the proposed message if the node has enough resources and n is in the preference list. If node m prefers n to the current accepted vNFs, these accepted vNFs may be rejected to accept n. Otherwise, it sends the "reject" message to n and deletes n and all vNFs n', $n \succ_m n'$ from its preference list.
   We denote $|list|^p$ as the total amount of resource type p of all accepted vNFs of node m.
3) Return to Step 1 until receiving a stop message or the preference list be empty.

In both distributed procedures, the stop message is received from a special agent, which can detect the quiescence of messages exchanged between vNFs and nodes.

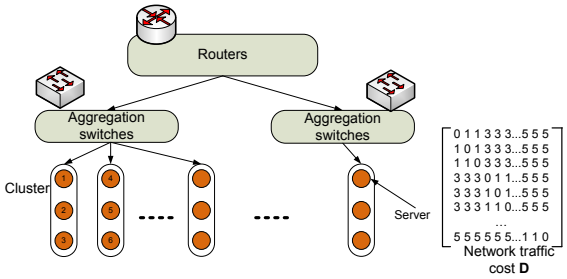
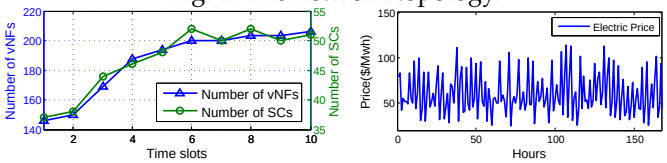


Fig. 4: The network topology.



(a) Trace of workload. (b) Electric price.

Fig. 5: Trace of workload and electric price.

7 Simulation and Numerical Results

In this section, we provide numerical settings and results to validate the efficacy of our proposed methods compared with other state-of-the-art approaches.

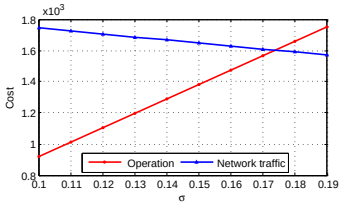
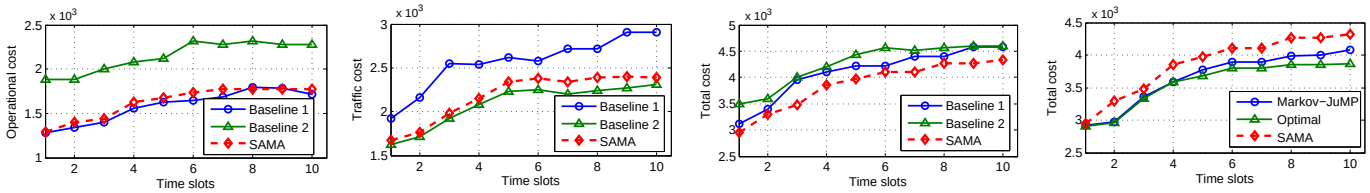


Fig. 6: Impacts of weight parameter σ on the cost.

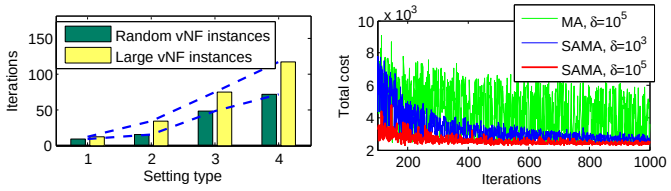
7.1 Simulation setup

We consider three types of vNFs whose configurations are set according to four representative instances of middleboxes [4], including firewall, IPSEC, intrusion detection



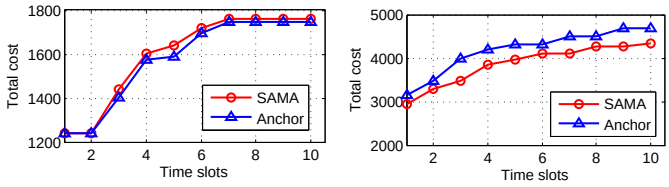
(a) Evaluation of the operational cost (b) Evaluation of the network traffic cost. (c) Evaluation of the total cost. (d) Comparison of the total cost.

Fig. 8: Evaluation of the cost incurred in the system within 10 time slots.



(a) Evaluation of fast convergence of Algorithm 4 (Setting type 1: N=200, M=120, type 2: N=400, M=220, type 3: N=600, M=350, type 4: N=1000, M=520). (b) Evaluation of convergence of MA and SAMA.

Fig. 7: Evaluation of the convergence.



(a) Evaluation of allocation with single vNF per SC. (b) Evaluation of allocation with multiple vNFs per SC.

Fig. 9: Comparison between SAMA and the Anchor framework.

systems (IDS) and WAN optimizer instances, as shown in Table 3.

In terms of the monetary weights, α, β are set to 0.1. By comparing operational cost with traffic cost in varying effect of σ as shown in Fig. 6, we observe that the network traffic cost is higher than operational cost when $\sigma < 0.173$, and vice version. Therefore, by choosing any value of $\sigma < 0.173$, we can emphasize the importance of the network traffic cost. It is clear that when σ is very small e.g., 0.1, we will have the traffic cost significantly dominate the operational cost. On the other hand, in this simulation, we would like to have a more balanced representation with $\sigma = 0.14$, where traffic cost still dominates the operational cost, but not significantly to avoid an extreme case. Of course, the network operator can set any value of σ lower than 0.173 to emphasize the network traffic cost, and vice versa as shown in Fig. 6. The power of active node Q_i is set 250 Watts uniformly, similar to [43]. We equally set the priority of all resources, where $\omega_p = 1, \forall p \in \mathcal{P}$. We also simulate our work with the configurations of vNFs based on the standard extra large instance on Table 3. Finally, we create the distance matrix D of the network for the system based on the topology in [34]. By using fast network simulation setup (FNSS) [44], we create the network topology with the values in range from 1 to 5, as shown in Fig. 4. Inside the cluster of servers, we set the link cost to 1 and the connection between two nodes inside the aggregation switches is set to 3. The highest cost is set to 5 for the connections going through the core routers. Furthermore, in order to illustrate the efficiency of SAMA, we evaluate our proposed method in each time slot with fixed parameter settings and over a long term with dynamic settings. For workload, we use the increasing trace workload [45] within 10 time slots to evaluate the impact of workload on the total cost of our proposed method. Moreover, we use the trace of electricity price in one week from [46] for our simulation, as shown in Fig. 5b in order to conduct the efficiency of reducing the total cost in OPNET corresponding to dynamic settings.

7.2 Results

In order to evaluate the convergence, optimality, and impact of SAMA, we compare the proposed mechanisms with three following baselines and prior approaches:

- Baseline 1: This baseline only uses a consolidation policy to place vNFs that optimizes the number of active nodes in the assignment.
- Baseline 2: This baseline only relies on the network traffic policy that optimizes the network traffic cost incurred in the system.
- Optimal: To quantify the gap between SAMA and the optimal solution, we compare SAMA to the optimal baseline that is solved by the JuMP solver [47]. Note that due to the inherent complexity of the optimal vNF placement problem, the time complexity of the JuMP solver turns out to be exponential in the large-scale problem.
- Markov-JuMP: Recall that the subproblem **P1** is solved by the matching approach that obtains a close optimal solution. To quantify the gap between the optimal solution (using the JuMP solver) and the matching approach, we compare SAMA to the baseline that is combined MA method and JuMP solver.
- Anchor: We also compare our approach with a matching-based consolidation algorithm, named Anchor in [40].

Convergence. We run Algorithm 4 and Algorithm 1 to evaluate the convergence of the matching and SAMA algorithms, respectively. The results in Fig. 7a show fast convergence in both simulation settings: the random instance size and the large instance size, in case of Algorithm 4. With the large vNF instances, there are many SCs that could not be deployed entirely into the same node due to lack of resources. Therefore, there are more rejected vNFs in each matching iteration. It also results in the number of

iterations increasing, but it does not increase exponentially with a large number of nodes M and the number of vNFs N. The convergence is not quite sensitive to the values of N and M, which shows the advantage of our matching approach. With the advantage of fast convergence, our mechanism impacts efficiently to the practical network consisting of thousands of nodes and vNFs. Hence, matching approaches can be applied to the environment of heterogeneous nodes and workload.

We next evaluate the convergence of Algorithm 1 with different values of δ and compare it to MA based approach. In theory, to approach the optimal solution of OPNET, we can set $\delta \rightarrow \infty$ as mentioned in Section 4. In practice, we select a value of δ that is large enough to limit the proposed algorithm that can converge within 1000 iterations. We explore the effect of δ by conducting an experiment as shown in Fig. 7b. For a larger value of δ, SAMA converges more closely to the optimal solution (i.e., the optimal gap is smaller). The total cost of OPNET decreases when the value of δ increases. The minimum value of total cost can be obtained around 800 iterations with $\delta = 10^5$. Meanwhile, comparing to the standard MA approach, it still fluctuates within 1000 iterations, owing to a large number of feasible configurations.

Total incurred cost. We next compare SAMA with two baselines, Baseline 1 and Baseline 2, in 10 time slots, as shown in Fig. 8. Baseline 1 aims to consolidate the resource utilization of nodes and reduces the number of active nodes, hence its average operational cost tends to be the lowest. However, considering from time slots 7 to 9, the operational cost of SAMA is lower than Baseline 1 since the wear-and-tear cost increases in Baseline 1.

Meanwhile, only optimizing the traffic cost, all vNFs of each SC in Baseline 2 aim to occupy one node or a group of nodes that incurs the lowest traffic cost, but this policy utilizes a lot of active nodes. Although it appears to be the worst in the first comparison case, Baseline 2 obtains the best result, when only traffic cost is considered, as shown in Fig. 8b.

By dynamically controlling the operational and network traffic costs, SAMA obtains the lowest total cost in all considered time slots as shown in Fig. 8c. Specially, when the workload is changed, Baseline 1 and Baseline 2 show the drawback in optimizing the network traffic cost and operational cost, respectively. During 10 time slots, SAMA reduces 19.1% and 9.28% of the total cost compared with the Baseline 1 and Baseline 2, respectively, as seen in Fig. 8c. It also demonstrates that jointly controlling the operational and the network traffic is efficient and significant in vNF placement problem. Furthermore, we illustrate the gap between the Optimal, Markov-JuMP and SAMA in Fig. 8d. Since each iteration invokes the matching algorithm to solve subproblem **P1** that does not guarantee an optimal result for the outcome, the total cost of SAMA still has a small gap that strictly follows Optimal and Markov-JuMP.

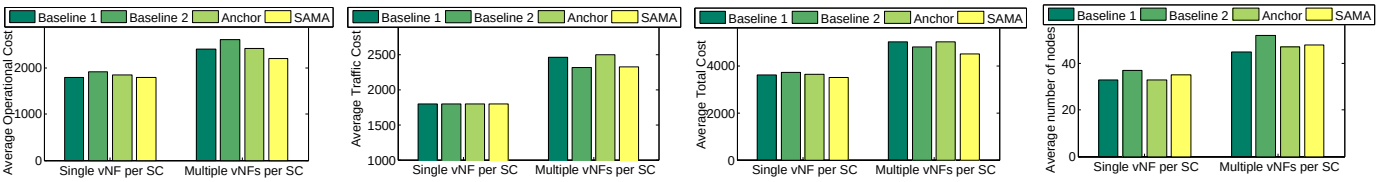
Comparison between SAMA and the Anchor framework. The Anchor framework [40] is a related approach that focuses on resource consolidation in datacenters using a matching game. However, the authors in [40] did not explain clearly how to build the preference lists for both sites in the matching method. Thus, we make the preference lists for both vNFs and nodes based on the Best fit algorithm (similar to the method discussed in Section 6.2).

In the first case with a single vNF instance in an SC, the total cost of SAMA and Anchor looks similar during 10 time slots, as shown in Fig. 9a, since the network traffic cost of all virtual links are the same during that period and the objective of OPNET mainly optimizes the operational cost. However, Anchor does not consider wear-and-tear costs that make the total cost increase over a long term (which is shown clearly in the next evaluation). In the second case, we compare SAMA and Anchor with multiple vNF instances in an SC. The Anchor framework ignores the interconnection between vNFs in each SC, which makes the total cost higher than SAMA. This causes the gap between SAMA and Anchor, as shown in Fig. 9b.

Comparison between SAMA and others in the long-run simulations. To characterize the effect of our proposed method in dynamic workload and electricity price settings, we evaluate our method over a long term. We illustrate clearly that our method outperforms others not only in a short term but also over a long term consideration. As mentioned in the assumption of Section 3, we use a discrete-time model, which has a time period (e.g., an hour) of interest. Then, we set the dynamic value of the price (α) to follow this trace and measure the efficiency of our method in long-term average operational cost, traffic cost, total cost and number of active nodes. We compare the results of four approaches (Baseline 1, Baseline 2, Anchor and SAMA) based on two kinds of SCs (single vNF per SC and multiple vNFs per SC) to reflect the efficiency of our method. Note that: our method does not guarantee the optimal solution over long term due to lack of information of workload in future.

In the case of a single vNF instance per SC, there are not many differences between approaches, as shown in Fig. 10. Because the network traffic cost of all virtual links are the same, and only the number of active nodes impacts the incurred total cost. The difference of each approach is depicted distinctly in the complex case of multiple vNFs per SC. In detail, Baseline 1 and Anchor utilize less number of active nodes comparing to others, as shown in Fig. 10d, so that their operational costs are also reduced. However, these approaches do not consider the wear-and-tear costs, which over a long term, make their operational costs increase more than SAMA, as illustrated in Fig. 10a. In contrast, Baseline 2 optimizes the traffic cost that is sketched in Fig. 10b. Conducting both energy cost and wear-and-tear cost in the operational cost, our method can reduce costs by about 8.7% in operational cost compared to Baseline 1. About the long-term traffic cost, SAMA consumes nearly the same network traffic cost as that of Baseline 2. Consequently, SAMA generates the lowest long-term cost compared to others.

Impact of wear-and-tear overhead. Finally, we evaluate the impact of wear-and-tear cost of SAMA and compare it with Baseline 1, Baseline 2 and Anchor. Baseline 1 and Anchor only reduce active nodes in deploying vNFs without considering wear-and-tear cost. Therefore, at each time slot, they pack vNFs into the smallest subset of active nodes and turn-off redundant physical nodes. Fig. 11 shows the highest wear-and-tear cost of Baseline 1 and Anchor. Regarding



(a) Long-term average for the operational cost. (b) Long-term average for the traffic cost. (c) Long-term average for the incurred total cost. (d) Long-term average for the number of active nodes.

Fig. 10: Comparison between SAMA and others in the long-run simulation.

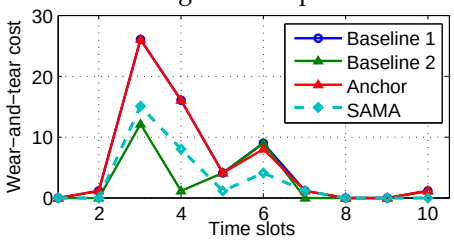


Fig. 11: Evaluation of wear-and-tear overhead.

Baseline 2, it seems to have the smallest wear-and-tear cost, but the varying wear-and-tear cost of Baseline 2 looks more fluctuated than SAMA. Since Baseline 2 relies on the vNF traffic cost, it does not optimize the active node set, which directly impacts wear-and-tear cost in the system. Our proposed method, SAMA, reduces this cost by about 27% compared to Baseline 1 during 10 time slots.

8 Conclusion

In this paper, we have studied the problem of joint operational and network traffic cost, OPNET, in the environment of heterogeneous nodes and diverse proprietary network appliances. We have formulated OPNET as the combinatorial NP-hard problem and designed a method to solve it, named SAMA. SAMA combines the MA method and many-to-one matching game to find a close-to-optimal solution, where the outcome has a small gap with the optimal solution. Furthermore, to implement on the practical NFV architectures that support centralized and distributed manners, we first have investigated the centralized approach that the controller can handle and compute an optimal allocation scheme in each time slot. We then have designed a distributed matching algorithm to decentralize the calculation to each physical node. To evaluate the efficiency of SAMA, we have compared our method with current methods that only consider traffic cost or operational cost. Simulation results show that SAMA can reduce the total cost by up to 19% compared to the existing non-coordinated approaches.

References

[1] NFV. [Online]. Available: https://portal.etsi.org/nfv/nfv_white_paper.pdf

[2] Service function chaining. [Online]. Available: https://tools.ietf.org/html/draft-boucadair-sfc-framework-02

[3] Cloud NFV White Paper.

[4] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, "Design and implementation of a consolidated middlebox architecture," in *Proceeding of 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, 2012, pp. 323–336.

[5] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing costs on service chain placement in network functions virtualization," in *IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, Nov 2015, pp. 191–197.

[6] Reducing costs by consolidation strategies. [Online]. Available: http://www.oracle.com/us/products/servers-storage/servers/sparc-enterprise/reducing-costs-wp-075962.pdf

[7] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of vDPI functions in NFV infrastructures," in *1st IEEE Conference on Network Softwarization (NetSoft)*, April 2015, pp. 1–9.

[8] A. Beloglazov and R. Buyya, "OpenStack neat: A framework for dynamic consolidation of virtual machines in OpenStack clouds–A blueprint," *Cloud Computing and Distributed Systems (CLOUDS) Laboratory*, 2012.

[9] M. Bagaa, T. Taleb, and A. Ksentini, "Service-aware network function placement for efficient traffic handling in carrier cloud," in *IEEE Wireless Communications and Networking Conference (WCNC)*, April 2014, pp. 2402–2407.

[10] S. Gu, Z. Li, C. Wu, and C. Huang, "An efficient auction mechanism for service chains in the NFV market," in *Proceedings of IEEE INFOCOM, San Francisco, CA, USA*, 2016.

[11] Service chains with vSRX. [Online]. Available: http://www.juniper.net/techpubs/en_US/vsrx15.1x49/topics/concept/security-vsrx-contrail-service-chains.html

[12] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," in *Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, vol. 11, 2011, pp. 24–24.

[13] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for NFV chaining in packet/optical data centers," in *European Conference on Optical Communication (ECOC)*, Sept 2014, pp. 1–3.

[14] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *3rd IEEE International Conference on Cloud Networking (CloudNet)*, Oct 2014, pp. 7–13.

[15] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1888–1906, Fourth 2013.

[16] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang, "An untold story of middleboxes in cellular networks," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, 2011, pp. 374–385.

[17] Y. Li and M. Chen, "Software-Defined Network Function Virtualization: A Survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[18] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, 2013, pp. 27–38.

[19] Y. Zhang and N. Ansari, "Heterogeneity aware dominant resource assistant heuristics for virtual machine consolidation," in *IEEE Global Communications Conference (GLOBECOM)*, Dec 2013, pp. 1297–1302.

[20] F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller, and J. Lawall, "Entropy: a consolidation manager for clusters," in *Proceedings of the ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, 2009, pp. 41–50.

[21] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for NFV chaining in packet/optical datacenters," *Journal of Lightwave Technology*, vol. 33, no. 8, pp. 1565–1570, April 2015.

[22] 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 *Proceedings of the 4th workshop on All Things Cellular: Operations, Applications, & Challenges*, 2014, pp. 33–38.

[23] A. Basta, A. Blenk, M. Hoffmann, H. J. Morper, K. Hoffmann, and W. Kellerer, "Sdn and nfv dynamic operation of lte epc gateways

for time-varying traffic patterns," in *International Conference on Mobile Networks and Management*. Springer, 2014, pp. 63–76.
[24] Network functions virtualisation (nfv);service quality metrics. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/010/01.01.01_60/gs_NFV-INF010v010101p.pdf
[25] C. Chappell, "Deploying virtual network functions: The complementary roles of tosca and netconf/yang," 2015.
[26] M. Lin, A. Wierman, L. L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 5, pp. 1378–1391, 2013.
[27] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: a scalable and flexible data center network," in *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, 2009, pp. 51–62.
[28] D. Meisner, B. T. Gold, and T. F. Wenisch, "PowerNap: Eliminating server idle power," *SIGARCH Comput. Archit. News*, vol. 37, no. 1, pp. 205–216, Mar. 2009. [Online]. Available: http://doi.acm.org/10.1145/2528521.1508269
[29] H. Qian and D. Medhi, "Server operational cost optimization for cloud computing service providers over a time horizon," in *Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*. USENIX Association, 2011, pp. 4–4.
[30] S. Ren and M. A. Islam, "Colocation demand response: Why do I turn off my servers," in *11th International Conference on Autonomic Computing (ICAC 14)*, 2014, pp. 201–208.
[31] N. H. Tran, D. H. Tran, S. Ren, Z. Han, E. N. Huh, and C. S. Hong, "How geo-distributed data centers do demand response: A game-theoretic approach," *IEEE Transactions on Smart Grid*, vol. 7, no. 2, pp. 937–947, March 2016.
[32] F. Ma, F. Liu, and Z. Liu, "Multi-objective optimization for initial virtual machine placement in cloud data center," *J. Infor. and Computational Science*, vol. 9, no. 16, 2012.
[33] S. K. Mandal and P. M. Khilar, "Efficient virtual machine placement for on-demand access to infrastructure resourcesin cloud computing," *International Journal of Computer Applications*, vol. 68, no. 12, pp. 6–11, April 2013, full text available.
[34] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of IEEE INFOCOM*, March 2010, pp. 1–9.
[35] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," in *Proceedings of IEEE INFOCOM*, March 2010, pp. 1–9.
[36] Y. Gu, W. Saad, M. Bennis, M. Debbah, and Z. Han, "Matching theory for future wireless networks: Fundamentals and applications," *IEEE Communications Magazine*, vol. 53, no. 5, 2015.
[37] M. Baiou and M. Balinski, "Erratum: The stable allocation (or ordinal transportation) problem," *Mathematics of Operations Research*, vol. 27, no. 4, pp. 662–680, 2002.
[38] S. Martello and P. Toth, *Knapsack problems*. Wiley New York, 1990.
[39] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, pp. 9–15, 1962.
[40] H. Xu and B. Li, "Anchor: A versatile and efficient framework for resource management in the cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1066–1076, June 2013.
[41] ——, "Egalitarian stable matching for vm migration in cloud computing," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2011, pp. 631–636.
[42] I. Brito and P. Meseguer, "Distributed stable matching problems," in *Principles and Practice of Constraint Programming-CP 2005*. Springer, 2005, pp. 152–166.
[43] Green IT: Making the Business Case. [Online]. Available: http://www.cognizant.com/InsightsWhitepapers/Green-IT-Making-the-Business-Case.pdf
[44] Reducing costs by consolidation strategies. [Online]. Available: http://fnss.github.io/
[45] Facebook, "Open sourcing pue, wue dashboards," https://code.facebook.com/posts/272417392924843/open-sourcing-pue-wue-dashboards.
[46] US Federal Energy Regulatory Commission. [Online]. Available: http://www.ferc.gov/
[47] JuMP. [Online]. Available: http://jump.readthedocs.org/en/latest/index.html

Chuan Pham received the BS degree from HCM City University of Transport and MS degree from University Of Science HCM City in 2004 and 2008,respectively. After graduation, he has been a lecturer in the Department of Computer Science, HCM City University of Transport. In 2013, he was awarded scholarship for his graduate study at Kyung Hee University, where he is currently working toward Ph.D. degree in Computer Science and Engineering.

Nguyen H. Tran received the BS degree from Hochiminh City University of Technology and Ph.D. degree from Kyung Hee University, in electrical and computer engineering, in 2005 and 2011, respectively. Since 2012, he has been an assistant professor in the Department of Computer Science and Engineering, Kyung Hee University.

Shaolei Ren received his B.E., M.Phil. and Ph.D. degrees, all in electrical engineering, from Tsinghua University in 2006, Hong Kong University of Science and Technology in 2008, and University of California, Los Angeles, in 2012, respectively. From 2012 to 2015, he was with Florida International University as an Assistant Professor. Since July 2015, he has been an Assistant Professor at University of California, Riverside.

Walid Saad (S07M10) received the B.E. degree in computer and communications engineering from the Lebanese University, Beirut, Lebanon, in 2004; the M.E. degree in computer and communications engineering from the American University of Beirut, Beirut, in 2007; and the Ph.D. degree from the University of Oslo, Oslo, Norway, in 2010. Since August 2014, he has been an Assistant Professor with the Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University (Virginia Tech), Blacksburg, VA, USA.

Choong Seon Hong received the BS and MS degrees in electronic engineering from Kyung Hee University, Seoul, Korea, in 1983 and 1985, respectively, and the Ph.D. degree at Keio University in March 1997. In 1988, he joined KT, where he worked on Broadband Networks as a member of the technical staff. In September 1993, he joined Keio University, Japan. He had worked for the Telecommunications Network Lab at KT as a senior member of the technical staff and as a director of the networking research team until August 1999. Since September 1999, he has been working as a professor of the Department of Computer Science and Engineering, Kyung Hee University.