Distributed Approach to Adaptive VNF Manager Placement Problem

Mao-Jung Chiang and Li-Hsing Yen
Department of Computer Science, College of Computer Science, National Chiao Tung University, Hsinchu, Taiwan.
Email:cmeo5xmm@gmail.com, lhyen@nctu.edu.tw

***Abstract*—Network function virtualization (NFV) has been a promising approach to flexible and scalable deployment of network services. In NFV management and orchestration (NFV-MANO) architectural framework, VNF managers (VNFMs) should be deployed to manage the lifecycle of virtualized network functions (VNFs). VNFM placement problem (MPP) is to deploy VNFMs that minimizes overall operational cost while meeting performance requirements. The only existing approach to MPP is centralized and does not well adapt to network dynamics (e.g., VNFM failures, ups and downs of VNF instances, etc.) We leverage game theory to achieve distributed solutions to the MPP, which are self-adaptive in the sense that each VNFM locally and autonomously adapts to network dynamics without a central control. Simulation results show the proposed approaches can adapt to network dynamics and have lower total cost than the counterpart in large-scale NFV systems.**

***Index Terms*—Distributed systems, network function virtualization, virtualized network function manager placement, game theory, potential games**

I. Introduction

In traditional telecommunication network, operators need to physically deploy specialized equipments or upgrade network infrastructure when new network services are offered to their customers. This practice is inflexible and not cost-effective because network services usually need dedicated network hardware, which is hard to reuse or augment new functionality on it. Consequently, deploying new services usually leads to large in both capital expenditures (CPEX) and operating expenditures (OPEX). Different from traditional network, network function virtualization (NFV) [1] promises to reduce CPEX and OPEX by leveraging virtualization technology to provide a new way to create and operate networking services. NFV decouples network functions from physical hardware and provides these functions as software applications running on commodity servers. As such, this approach significantly reduces the cost of purchasing dedicated hardware. In addition, apart from improvement on operational and capital efficiencies, NFV also reduces power usage and makes networks more scalable and flexible.

NFV management and orchestration (NFV-MANO) architectural framework proposed by ETSI [2] consists of three main components: Virtualized Network Functions (VNFs), Network Functions Virtualization Infrastructure (NFVI) and NFV-MANO. VNFs are software implementations of traditional middleboxes, such as firewall, deep packet inspector, and load balancer, running as network functions that can be deployed on a NFVI. NFVI combines hardware with software resources to build NFV environment. It may be distributed with multiple geographically-distributed locations called NFVI Point of Presences (NFVI-PoPs) for resource access.

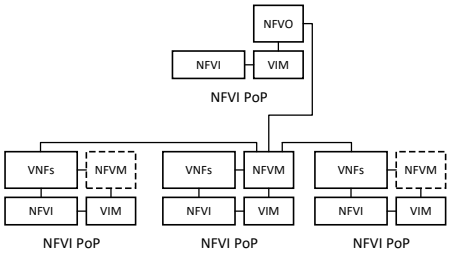


Fig. 1. NFV architectural in geo-distributed NFVI PoPs

NFV-MANO [3] is for orchestration and resources management on NFVI and is in charge of controlling lifecycle of network services. It is composed of three functional blocks: NFV Orchestrator (NFVO), VNFs manager (VNFM) and Virtualized Infrastructure Manager (VIM). These function blocks communicate with each other and with other components in the NFV architectural framework. NFVO handles on-boarding of network services and VNF packages, lifecycle management of network services, and validation and authorization of NFVI resource requests. VNFM is responsible for lifecycle management of VNF instances, including instantiation, maintenance (e.g., scaling, updating, upgrading) and termination of VNFs. VIM controls and manages the compute, storage, and network resources of NFVI. It also collects performance information and reports resources failure.

The overall operational cost of NFV service deployment consists of resource (compute and bandwidth) cost, energy cost, and possible payments. The operation cost is location-dependent and various over time. This study focuses on how to reduce the operation cost by a dynamic placement of VNFMs. The cost reduction is significant particularly in large-scale distributed NFV systems, as shown in Figure 1. To the best of our knowledge, the work by Mohammad et al. [4] is the only study toward VNFM placement problem (MPP). Their solution is based on Tabu search meta-heuristic [5]. Because it is a centralized approach performed by NFVO, a new deployment has to be found from scratch when traffic condition differs significantly or some componenet (VNFM, NFVI, etc.) fails after a deployment.

In this paper, we proposed a game-theoretic approach to MPP, where VNFMs NFVI-PoPs (rather than NFVO) autonomously determine whether to activate themselves and, if they do, which VNF instances are under their managements. This approach is fully distributed and self-adaptive: the set of active VNFMs can locally adapt to network and service dynamics, which serves a way to cope with component failures. We propose two versions of the approach, one converges fast while the other generally yield deployment of lower cost.

The remainder of this paper is organized as follows: Sec. II presents related work and background. Sec. III details the proposed approach to MPP. Section IV presents performance evaluations on the proposed approaches. The last section concludes this work.

II. Related Work and Background

A. Related Work

Many researches have been done toward VNF placement and service chaining problems, which aim at finding the number of VNF instances needed for each network service and the commodity servers where to run these VNF instances [6], [7], [8], [9], [10]. However, little effort has been put on MPP. To the best of our knowledge, only Abu-Lebdeh et al. [4] had studied this problem. They employed Tabu search algorithm to solve the VNFM placement problem in both static and dynamic ways. However, Tabu is a randomized approach running in iterations. In each iteration, it randomly generates a large number of possible solutions, called move list, and selects the best move. Best moves will be kept. When the the algorithm terminates, the best solution kept in the list of best moves will be selected.

Our study differs from [4] in that our approach is a distributed mechanism which locally and autonomously adapt to NFV system dynamics (ups and downs of VNFMs and NFV instances). In contrast, people has to rerun the Tabu-based approach periodically or whenever network condition changes significantly.

B. System Model

We follow the system model of [4] with some modifications to make the following assumptions: 1) NFVO and VNFM can be implemented separately. 2) Each NFVI-PoP has a VNFM which can be active or inactive. 3) A VNF instance and its Element Management (EM) are bound together and deployed at the same NFVI-PoP. 4) A VIM manages the resources of the NFVI-PoP where it locates. 5) The communication between different functional blocks flows over the same network links as regular traffic.

In NFV, the network services and corresponding VNFs can be instantiated whenever needed. Therefore, the number of VNF instances and their locations vary dynamically. Each VNF instance must be managed by an active VNFM while an active VNFM can serve multiple VNF instances as long as the number of VNF instances does not exceed the capacity of the VNFM. While VNF instances can be of different types, VNFMs can manage any type of VNF(s) as they are typically generic.

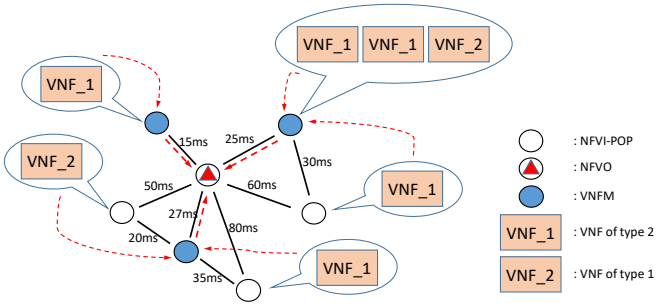


Fig. 2. An example of VNFM deployment

In this study, we assume that the locations of VIMs and NFVO are fixed while VNF instances are scattered over on geographically-distributed NFVI-PoPs. Therefore, an VNFM needs to communicates with the VNF instances under its management, Element Managers (EMs), VIMs and NFVO in the system over Wide Area Network (WAN) links. A possible solution to MPP should ensure that all the communications meet respect VNF-specific delay constraints set by NFVO. The solution should also meet VNFM capacity constraint in terms of the number of managed VNF instances and bandwidth capacity constraint on all communication links.

Consider a NFV architecture with 7 NFVI-PoPs shown in Figure 2. Each NFVI-PoPs may host one or more VNF instances and one VNFM. We assume the location of the NFVO is already given and a VNFM can manage 20 VNF instances at a time. Delay constraint between VNFM and NFVO is 40 ms for all VNF instances. This figure shows a feasible solution that the delay and the capacity constraints of VNFMs are all met.

C. Problem Formulation

The MPP can be formally defined as follows. Given the location of the NFVO and configuration of VNF instances over geographically-distributed NFVI-PoPs, determine the set of NFVI-PoPs to deploy VNFMs and the set of managed VNF instances for each VNFM with the minimal total cost such that all VNF instances are managed and all delay, bandwidth, and capacity constraints are met. We aim at designing a self-adaptive distributed solution to MPP.

Table I shows notations used in this paper. We additionally define two decision variables below.

$$x_p = \begin{cases} 1, & \text{if the VNFM at } p \in P \text{ is active,} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

$$y_{j,p} = \begin{cases} 1, & \text{if } j \in V \text{ is assigned to the VNFM} \\ & \text{at } p \in P. \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

The total operational cost consists of bandwidth cost and compute cost. The bandwidth cost C^{lif} counts the cost of net-

TABLE I
NOTATIONS

	NFVI
$G(P, E)$	NFVI Graph G with P the set of NFVI-PoPs and E the set of links
$\gamma_{p,q}$	Capacity of link $(p, q) \in E$
$\delta_{p,q}$	Delay of link $(p, q) \in E$
c_p^{com}	Cost of compute resource at $p \in P$
$c_{p,q}^{\text{net}}$	Bandwidth cost over link $(p, q) \in E$
	NFVO
$h_p \in \{0, 1\}$	$h_p = 1$ if NFVO is placed at $p \in P$
	VNFM
M	Set of active VNFMs with $\lvert M\rvert = m$
n_p	Capacity of the VNFM at $p \in P$
	VNF
V	Set of VNF instances with $\lvert V\rvert = v$
$l_{j,p} \in \{0, 1\}$	$l_{j,p} = 1$ if VNF instance j is placed at $p \in P$ and $l_{j,p} = 0$ otherwise
φ_j	Maximum permissible delay between VNF instance j and the VNFM managing it
ω_j	Maximum permissible delay between the NFVO and the VNFM managing VNF instance j
	Network Traffic
$u_j^{O,M}$	Bandwidth consumed between NFVO and the VNFM that manages VNF instance j
$u_j^{O,I}$	Bandwidth consumed between NFVO and VIM concerning VNF instance j
$u_j^{M,I}$	Bandwidth consumed between VNFM and VIM concerning VNF instance j
$u_j^{M,V}$	Bandwidth consumed between VNFM and EM/VNF instance j

work bandwidth consumed in the communication of managing lifecycle of all VNF instances in the system.

$$C^{\text{lif}} = \sum_{(p,q)\in E} \left(B^{\text{lif}}(p,q) + B^{\text{lif}}(q,p)\right) c_{p,q}^{\text{net}}, \tag{3}$$

where

$$\begin{aligned} B^{\text{lif}}(p,q) = \sum_{j\in V} \Big\{ & y_{j,p} h_q u_j^{O,M} + y_{j,p} l_{j,q} \\ & \left(u_j^{M,V} + u_j^{M,I}\right) + l_{j,p} h_q u_j^{O,I} \Big\}. \end{aligned} \tag{4}$$

The compute cost C^{com} is the overall energy cost of all active VNFMs.

$$C^{\text{com}} = \sum_{p\in P} x_p c_p^{\text{com}}. \tag{5}$$

The objective of MPP is to minimize the weighted sum of the aforementioned costs. Let ε and θ be two weighting factors. Then the objective function can be expressed as follows:

$$\min \ \varepsilon C^{\text{lif}} + \theta C^{\text{com}}. \tag{6}$$

Next, we discuss the constraints of MPP. Each VNF instance should be managed by one VNFM, i.e.,

$$\sum_{p\in P} y_{j,p} = 1, \quad \forall j \in V. \tag{7}$$

The following constraint indicates that VNF instance j cannot be managed by the VNFM at NFVI-PoP p unless the VNFM is active:

$$y_{j,p} \le x_p, \quad \forall j \in V, p \in P. \tag{8}$$

Capacity constraint demands that the number of VNF instances assigned to each VNFM does not exceed its capacity:

$$\sum_{j\in V} y_{j,p} \le n_p, \quad \forall p \in P. \tag{9}$$

A VNFM is active only when it manages at least one VNF instance, as indicated in (10):

$$x_p \le \sum_{j\in V} y_{j,p}, \quad \forall p \in P. \tag{10}$$

Delay constraints of each VNF instance are defined by (11) and (12). Eq. (11) is for the delay between VNF instance j and the VNFM managing it.

$$\left(y_{j,p} l_{j,q} + y_{j,q} l_{j,p}\right) \delta_{p,q} \le \varphi_j, \quad \forall j \in V, (p,q) \in E. \tag{11}$$

Eq. (12) is for the delay between the NFVO and the VNFM managing VNF instance j.

$$\left(y_{j,p} h_q + y_{j,q} h_p\right) \delta_{p,q} \le \omega_j, \quad \forall j \in V, (p,q) \in E. \tag{12}$$

Eq. (13) ensures that the bandwidth consumed on each link does not exceed the bandwidth capacity:

$$B^{\text{lif}}(p,q) + B^{\text{lif}}(q,p) \le \gamma_{p,q}, \quad \forall (p,q) \in E. \tag{13}$$

The last two constraints make sure that the outputs are all integers:

$$x_p \in \{0, 1\}, \quad \forall p \in P. \tag{14}$$

$$y_{j,p} \in \{0, 1\}, \quad \forall j \in V, p \in P. \tag{15}$$

III. The Proposed Approach

A. Game Mechanism

The basic idea is to exploit potential game theory to achieve distributed solutions to the MPP. Each VNFM acting as a player in a potential game unilaterally decides whether to activate itself and, if it does, the set of VNF instances under its management. As such, the objective function of MPP is decoupled to the utility function to each player in the game. While each player individually aims at maximizing its own utility, the system can benefit from high cost efficiency.

We propose two versions of the game designs. Version A has a fast convergence time and performs well in small-scale environment. On the other hand, Version B though more complicated can achieve better cost efficiency in large-scale network.

Let $P = \{1, 2, \ldots, n\}$ denote the player set. Each player p has a strategy set $S_p = C_{p1} \times C_{p2} \times \ldots \times C_{pv}$, where $C_{pj} = \{0, 1\}$, $j = 1, 2, \ldots, v$, indicates whether VNFM p manages VNF instance j. A strategy of player p is a v-tuple $\mathbf{c}_p = (c_{p1}, c_{p2}, \ldots, c_{pv}) \in S_p$, where $c_{pj} \in C_{pj}$. A *strategy profile* is an n-tuple $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n)$. Sometimes we may express $\mathbf{c}$ as $(\mathbf{c}_p, \mathbf{c}_{-p})$ where $\mathbf{c}_{-p} = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{p-1}, \mathbf{c}_{p+1}, \ldots, \mathbf{c}_n)$.

The utility function $u_p(\mathbf{c})$ for every player $p \in P$ governs p's behavior because p's objective is to maximize its utility by selecting a strategy that is its *best response* to other player's decisions. Formally, p's objective is $\max_{\mathbf{c}_p \in S_p} u_p(\mathbf{c}_p, \mathbf{c}_{-p})$.

Before presenting player's utility function, we introduce some auxiliary variables. Define $k_p(\mathbf{c}_p)$ to be an indicator variable indicating whether p's strategy $\mathbf{c}_p$ meets the capacity constraint as defined in (9). $k_p(\mathbf{c}_p) = 1$ if the number of VNF instances managed by VNFM p does not exceed VNFM p's capacity, and $k_p(\mathbf{c}_p) = 0$ otherwise. Formally,

$$k_p(\mathbf{c}_p) = \begin{cases} 1, & \text{if } \sum_{j=1}^{v} c_{pj} \leq n_p, \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

Let $N_p(\mathbf{c}_p)$ indicates whether the capacity of VNFM at location p is just full:

$$N_p(\mathbf{c}_p) = \begin{cases} 1, & \text{if } \sum_{j \in V} c_{pj} = n_p. \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

Let $f_{pj}(\mathbf{c})$ indicates whether $\mathbf{c}_p$ meets the bandwidth constraint (13). Let d_{pj}^{vm} and d_{pj}^{om} be two variables indicating whether $\mathbf{c}_p$ meets the two delay constraints (11) and (12), respectively.

In Version A, VNFM p can take strategy $\mathbf{c}_p$ as long as all the constraints are met. This is specified as a condition C_1:

$$C_1 \equiv f_{pj}(\mathbf{c}) d_{pj}^{vm} d_{pj}^{om} \sum_{k \in P} c_{kj} = 1.$$

C_1 is true only when $\sum_{k \in P} c_{kj} = 1$, which implies that no other VNFM also manages VNF instance j.

The definition of $u_p(\mathbf{c}_p, \mathbf{c}_{-p})$ for every $p \in P$ concerns the gain of $\mathbf{c}_p$ with respect to $\mathbf{c}_{-p}$. We define a *gain function* $g_{p,j}(\mathbf{c})$ to denote the amount of profit that VNFM p can earn by managing VNF instance j. Managing a VNF instance by the VNFM located at the same NFVI-PoP is always perferred because that can save the bandwidth cost of the communications between VNFM and VIM/EM. Therefore, if VNF instance j is located at p and p is able to manage it, VNFM p will get a positive gain value θc_p^{com} by managing j. If VNF instance j is located at $q \neq p$ and the capacity of q is already full or the delay between VNF instance j and the NFVO exceeds the delay bound, p can gain a positive value α by managing j. For other conditions, we define the gain to be the difference of cost before and after p decides to manage VNF instance j. The gain could be positive or negative. Let G_1 be the cost difference when the NFVO is located at the same NFVI-PoP as VNFM p. Let G_2 be the cost difference when VNF instance j is located at same NFVI-PoP as the NFVO. Let G_3 be the cost difference when the NFVO is located at a NFVI-PoP different from VNF instance j's and VNFM p's locations. Formally,

$$\begin{aligned} G_1 &= \theta \left(c_q^{\text{com}} - c_p^{\text{com}}\right) - \varepsilon \left(u_j^{M,V} + u_j^{M,I} - u_j^{O,M}\right) c_{p,q}^{\text{net}}, \\ G_2 &= \theta \left(c_q^{\text{com}} - c_p^{\text{com}}\right) - \varepsilon \left(u_j^{M,V} + u_j^{M,I} + u_j^{O,M}\right) c_{p,q}^{\text{net}}, \\ G_3 &= \theta \left(c_q^{\text{com}} - c_p^{\text{com}}\right) - \varepsilon \left(u_j^{M,V} + u_j^{M,I}\right) c_{p,q}^{\text{net}}. \end{aligned} \tag{18}$$

Then $g_{p,j}(\mathbf{c})$ can be defined as

$$g_{p,j}(\mathbf{c}) = \begin{cases} \theta\ c_p^{\text{com}}\ , & \text{if } C_1 \text{ and } l_{j,p} = 1, \\ \alpha\ , & \text{if } C_1 \text{ and } (l_{j,p} = 0,\ N_q(\mathbf{c}_q) = 1) \\ & \text{or } \omega_j < \delta_{q,o}, \\ G_1\ , & \text{if } C_1 \text{ and } l_{j,p} = 0, N_q(\mathbf{c}_q) = 0, \\ & h_p = 1 \text{ and } \omega_j \geq \delta_{q,o}, \\ G_2\ , & \text{if } C_1 \text{ and } l_{j,p} = 0, N_q(\mathbf{c}_q) = 0, \\ & h_q = 1 \text{ and } \omega_j \geq \delta_{q,o}, \\ G_3\ , & \text{if } C_1 \text{ and } (l_{j,p}, h_p, N_q(\mathbf{c}_q), h_q) = \mathbf{0} \\ & \text{and } \omega_j \geq \delta_{q,o}, \\ -\alpha\ , & \text{otherwise,} \end{cases} \tag{19}$$

where $\alpha > 0$ is a constant, q is the location of VNF instance j, and o is the location of the NFVO, with the assumption that $c_{p,q}^{\text{net}}$ is a constant for all $p, q \in P$. If the gain is positive, meaning that managing VNF instance j can lower overall cost, VNFM p is supposed to manage j. VNFM p will get a negative gain value $-\alpha$ if managing j does not meet one or more constants.

The utility of VNFM p is the summation of gain values over all VNF instances managed by it:

$$u_p(\mathbf{c}) = \begin{cases} 0\ , & \text{if } \mathbf{c}_p = \mathbf{0}, \\ k_p(\mathbf{c}_p) \sum_{j=1}^{v} c_{pj} g_{p,j}(\mathbf{c})\ , & \text{otherwise.} \end{cases} \tag{20}$$

Version B differs from Version A in that VNFMs have priorities in making decisions. In Version A, once VNFM p has managed VNF instance j, any other VNFM cannot manage j unless p fails. In Version B, a VNFM q can "rob" p of j provided that q's priority is higher than p's. The basic idea is: VNFMs with low computation costs usually can have more VNF instances than VNFSs with high computation costs. By giving these VNFMs opportunities to grab VNF instances from others, the whole system has a high probability to deactivate VNFMs with high computation costs.

Let $\nu_p(\mathbf{c}_p) = \sum_{j=1}^{v} c_{pj}$ be the total number of VNF instances managed by VNFM p. We take $\nu_p(\mathbf{c}_p)$ as p's priority value and redefine condition C_1 to be

$$\begin{aligned} C_1 \equiv (!\exists q \in P \backslash \{p\} : c_{qj} = 1, \nu_q(\mathbf{c}_q) \geq \nu_p(\mathbf{c}_p)) \text{ and} \\ f_{pj}(\mathbf{c}) d_{pj}^{vm} d_{pj}^{om} = 1. \end{aligned} \tag{21}$$

The second part of C_1 is the same as in Version A. The first part of C_1 is to ensure the absence of any VNFM q with priority value equal to or higher than p.

However, the utility of a player depends not only on its own strategy, but also the strategies of other players. Clearly, the game model assumes a player knows the current strategies of all other players when making a decision. This assumption demands communication synchronization in the system. Here we suppose NFVO play an important role in communication synchronization. NFVO collects the information about decisions of VNFMs and each VNFM can ask for current state of system before making a decision.

B. Proof of Convergence

The dynamics of the game are highly complex even for small-scale environment. It is theoretically possible that VNFMs as players keep changing their strategies and the game cannot end up with a stable result. The stability of such a dynamic game corresponds to a Nash equilibrium defined below.

Definition 1 (Nash equilibrium): Given a game $\Gamma = [P; \{S_i\}_{i=1}^n; \{u_i\}_{i=1}^n]$, a strategy profile $\mathbf{c}^* = (\mathbf{c}_1^*, \mathbf{c}_2^*, ..., \mathbf{c}_n^*)$ is a Nash equilibrium if $\forall i \in \{1, 2, ..., n\} : \forall \mathbf{c}_i \in S_i : u_i\left(\mathbf{c}_i^*, \mathbf{c}_{-i}^*\right) \geq u_i\left(\mathbf{c}_i, \mathbf{c}_{-i}^*\right)$.

We prove that the propose game always converges to a Nash equilibrium by showing that the game is an exact potential game [11].

For Version A, we assume that the game starts from the initial strategy profile where every VNFM manages no VNF instance at all. We can prove that this game is an exact potential game by showing an exact potential function for it:

$$\pi(\mathbf{c}) = \sum_{i \in P} u_i(\mathbf{c}). \tag{22}$$

It can be verified that $\pi(\mathbf{c})$ is an exact potential function of the game. The key point is when a player changes its decision, that change does not affect any other's utility. A game with this property is known as a self-motivated game. Since the strategy space of our game is finite and every finite exact potential game possesses a Nash equilibrium [11], the game eventually ends up with a Nash equilibrium.

IV. Experimental Results

We studied the performance of the proposed approach and the Tabu-based approach [4] through simulations. We considered both small-scale and large-scale NFV systems. The major performance metric for comparions was total operational cost. For the proposed adaptive approach, time to convergence and the ratio of affected VNFMs were also measured. Each simulation result was averaged over hundreds of trials with the same configuration.

A. Simulation Settings

We set weighting factors ε and θ to 1000 and 1, respectively, in the simulations. This setting was to balance compute cost and bandwidth cost. The compute cost at each NFVI-PoP (c_p^{com}) ranged from 1 to 2.3. Link delay between each pair of NFVI-PoPs ($\delta_{p,q}$ for all $p, q \in P$) ranged from 1 to 100 ms. The bandwidth cost of each link ($c_{p,q}^{\text{net}}$ for all $p, q \in P$) was in proportional to the consumed bandwidth and equaled \$0.5/GB. The capacity of each edge ($\gamma_{p,q}$ for all $p, q \in P$) was set to 10 Gbps. For simplicity, we assumed that the bandwidth capacity of any link is always sufficient.

We considered two types of VNFs: type one (T1) and type two (T2). Compared with T2 VNFs, T1 VNFs consumed more bandwidth and had lower delay tolerance. Table II lists the parameters of T1 and T2 VNFs. The type of a VNF in the simulation was randomly determined.

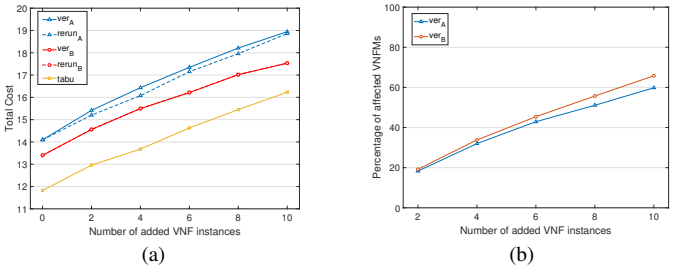
TABLE II
Experiment parameters for VNF types T1 and T2

Parameter	Unit	T1	T2
φ_j	ms	35	60
ω_j	ms	70	100
$u_j^{O,M}$	Mb/h	0.5	0.1
$u_j^{O,I}$	Mb/h	0	0
$u_j^{M,I}$	Mb/h	1.2	0.3
$u_j^{M,V}$	Mb/h	1.8	0.3

B. Small-Scale Scenario

We deployed 10 NFVI-PoPs and set the capacity of each VNFM (n_p) to 10. We ran Version A and Version B to get original results. After that, we dynamically added 2 to 10 VNF instances to the system. In Fig. 3, rerun_A, rerun_B, and tabu are the results of rerunning Version A, Version B, and the Tabu-based approach, respectively, from scratch for the new deployment. The results labeled with ver_A and ver_B were obtained by adaptive runs of Versions A and B, respectively. For adaptive runs, the original results were used as initial configurations of VNFMs before starting the responses to the new deployments.

The results show that Tabu-based approach had the lowest total cost than the others. rerun_B and ver_B had nearly the same cost while the results with rerun_A and ver_A were difficult. The reason is that Version B has a smaller solution space so running it from scratch or an adaptive run do not matter. The gap between rerun_A and ver_A is within an acceptable range. Fig. 3(b) shows the percentage of affected VNFMs in adaptive runs. Even when 10 VNF instances were added to the system, only around 60% VNFMs changed their settings.



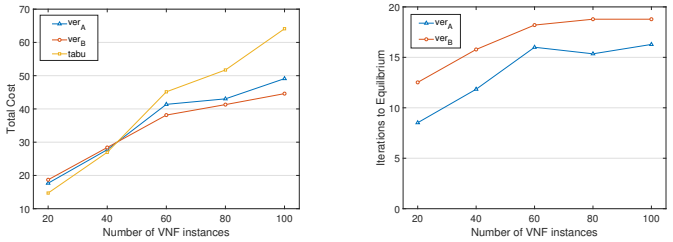
(a) (b)

Fig. 3. Results in small-scale scenario. (a) Total cost (b) Percentage of affected VNFMs

C. Large-Scale Scenario

We deployed 16 NFVI-PoPs and set the VNFM capacity (n_p) to 20 in the large-scale scenarios. Fig 4(a) shows how the costs of all different approaches changed with the number of VNF instances. When there were 40 or fewer VNF instances, the Tabu-based approach had the lowest total cost. However, as the number of VNF instances increased, its performance was overtaken by both Versions A and B. The reason is that the Tabu search is more likely to stuck in local optimum when the solution space expends as the number of VNF instances increases. Version B performed the best when more VNF instances were involved.

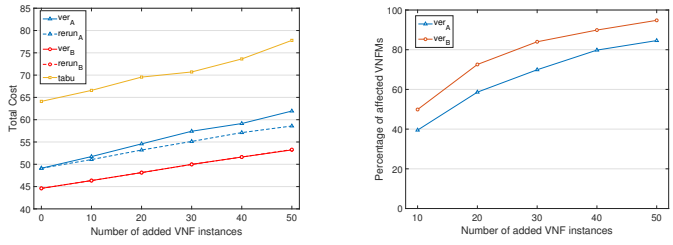
Figure 4 (b) shows the number of iterations (strategy changes or player's moves) before reaching a Nash equilibrium in both Versions A and B. We can see that the outperformance of Version B over Version A comes at the cost of larger convergence time.



(a) (b)

Fig. 4. Results in large-scale scenario. (a) Total cost (b) Percentage of affected VNFMs

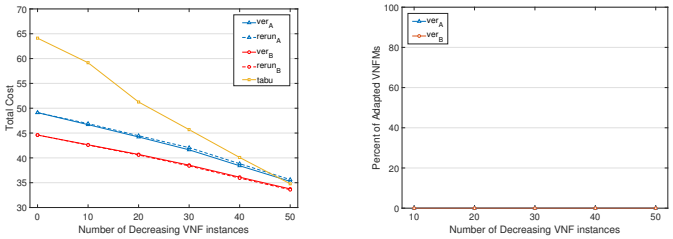
Figure 5 shows the results of adding 10 to 50 VNF instances to the system after VNFM deployment. As indicated in Figure 5(a), the Tabu-based approach had the highest total cost. Version B performed the best, regardless whether it was run from scratch or adaptively. Adaptive runs of Version A had costs only slightly higher than rerunning it. Its performance is lower than Version B. However, concerning the percentage of affected VNFMs, Version A performed better than Version B because an adaptive run of Version A generally affected fewer VNFMs than that of Version B as indicated in Figure 5(b).



(a) (b)

Fig. 5. Results of adding VNF instances to large-scale NFV system. (a) Total cost (b) Percentage of affected VNFMs

Figure 6 shows the results when 10 to 50 VNF instances were removed from the NFV system after VNFM deployment. Figure 6(a) shows the total cost of each approach. All approaches had a decreasing cost but the Tabu approach had the most dramatic decline. The reason is that Tabu can have a better performance with fewer VNF instances (small-scale NFV system). Besides, as shown in Figure 6(b), VNFMs basically did not change their strategies when the number of VNF instances decreased.



(a) (b) Cost

Fig. 6. Results of removing VNF instances from large-scale NFV system. (a) Total cost (b) Percentage of affected VNFMs

V. Conclusions

We have shown how to leverage game-theoretic model for the design of a distributed adaptive solution to the VNFM placement problem (MPP). We have proved the stability of the proposed solution, implying that it locally and autonomously adapts to the dynamics of the NFV environment. The simulation results show that the proposed solutions have lower total cost than the Tabu-based counterpart in a large-scale NFV environment. Two versions of our solutions were also compared experimentally. The results show that Version A has faster convergence time but higher total cost than Version B.

References

[1] ETSI, "Network functions virtualization-introductory white paper," *SDN and OpenFlow World Congress*, 2012. [Online]. Available: http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[2] ——, "Network functions virtualisation (NFV): Architectural framework," 2013.

[3] ——, "ESTI NFV management and orchestration: An overview," 2013. [Online]. Available: https://www.ietf.org/proceedings/88/slides/slides-88-opsawg-6.pdf

[4] M. Abu-Lebdeh, D. Naboulsi, R. Glitho, and C. W. Tchouati, "On the placement of VNF managers in large-scale and distributed NFV systems," *IEEE Trans. on Network and Service Management*, vol. 14, no. 4, pp. 875–889, Dec. 2017.

[5] F. Glover, "Tabu search: Part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[6] H. Moens and F. D. Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *10th Int'l Conf. on Network and Service Management (CNSM) and Workshop*, Nov. 2014, pp. 418–423.

[7] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Trans. on Network and Service Management*, vol. 13, no. 4, pp. 725–739, Dec. 2016.

[8] J. Gil Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, Sep. 2016.

[9] S. D'Oro, L. Galluccio, S. Palazzo, and G. Schembra, "Exploiting congestion games to achieve distributed service chaining in NFV networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 2, pp. 407–420, Feb. 2017.

[10] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, "Traffic-aware and energy-efficient vNF placement for service chaining: Joint sampling and matching approach," *IEEE Trans. on Services Computing*, pp. 1–1, 2018.

[11] D. Monderer and L. S. Shapley, "Potential games," *Games and Economic Behavior*, vol. 14, no. 1, pp. 124 – 143, 1996.