A^2VF: Adaptive Allocation for Virtual Network Functions in Wireless Access Networks

Pham Tran Anh Quang*, Abbas Bradai†, Kamal Deep Singh‡, Roberto Riggio§

*IRISA / University of Rennes 1, Campus de Beaulieu, 35042 Rennes, France
Email:{quang.pham-tran-anh}@irisa.fr

†Laboratoire Hubert Curien, Université de Saint-Etienne, Jean Monnet, F-42000, Saint-Etienne, France
Email:kamal.singh@univ-st-etienne.fr

‡XLIM, University of Poitiers, Poitiers, France
Email: abbas.bradai@univ-poitiers.fr

§ FBK CREATE, Via Alla Cascata 56/C, 38123 Povo, Trento, Italy
Email: rriggio@fbk.eu

***Abstract*—Network Function Virtualization (NFV) is deemed as a mean to simplify deployment and management of network and telecommunication services. With wireless access networks, NFV has to take into account the radio resources at wireless nodes in order to provide an end-to-end optimal virtual network function (VNF) allocation. This topic has been well-studied in existing literature, however, the effects of variations of networks over time have not been addressed yet. In this paper, we provide a model of the adaptive and dynamic VNF allocation problem considering VNF migration. Then we formulate the optimisation problem as an Integer Linear Programming (ILP) and provide a heuristic algorithm for allocating multiple service function chains (SFCs). The proposed approach allows SFCs to be reallocated so as to obtain the optimal solution over time. The results confirm that the proposed algorithm is able to optimize the network utilization while limiting the reallocation of VNFs which could interrupt services.**

I. Introduction

Network Function Virtualization (NFV) promises to reduce the cost of deploying and operating large network infrastructures. It provides the possibility to migrate complex network functions from dedicated hardware appliances to general purpose computing, storage, and networking solutions. This transition is also expected to provide significant benefits to the 5G mobile network architecture, by allowing flexible and scalable provisioning of new applications and network services (software vs. hardware development life-cycles). Moreover, since NFV allows multiple network services to share the same physical infrastructure, it enables new business models and/or economies of scale. In particular, the Network–as–a–Service (NaaS) business model is expected to play a pivotal role in 5G mobile networks, allowing Mobile Network Operators (MNOs) to tap into new revenue streams. Virtualization will allow MNOs to abstract their physical network infrastructure into service specific slices, possibly operated by different mobile virtual network operators (MVNOs) or over the top (OTT) providers. The envisioned vertical applications range from high–definition video delivery to machine–to–machine applications.

Virtualization and *adaptive* network service orchestration are two of the main technical enablers that will allow Infrastructure Providers (InPs) to cope with the diverse range of requirements that will characterize future applications and services. It is worth noticing that in this case, the InPs could be either a traditional MNO that decides to open-up its networks to third parties or a new actor in the value chain that focuses only on the deployment and operation of the network infrastructure (the InP could even be a consortia of MNOs). Although a rich body of literature exists on VNF placement [1], virtual network embedding [2], and component placement [3], most of these works focus on the problem of mapping an input virtual network request (often in the form of a VNF Forwarding Graph) onto a physical virtualized network substrate (often offering computational as well as networking resources). Nevertheless, most of these works either assume that on–boarded network services are not changed after they are mapped onto the substrate network or, when network service remapping is allowed, they do not consider the network service migration cost.

In this work, we address the adaptive allocation of Virtual Network Functions in mobile and wireless networks. We model the adaptive and dynamic VNF allocation problem considering also VNF migration. Then we formulate the optimisation problem as an Integer Linear Programming (ILP). As the optimisation problem is found to be NP-hard, we provide a heuristic algorithm for allocating multiple service function chains (SFCs), which can provide near-optimal solutions in polynomial time. We expect MVNOs to specify their requests in terms of a VNF Forwarding Graph. Such VNFs include all the typical elements of a mobile or wireless network plus the typical middleboxes found in this context, e.g. load–balancers, firewalls, and deep packet inspection devices. Note that how with mobile or wireless network elements we address both the radio (virtual base stations) and the core (virtual Evolved Packet Core) segments of the mobile network architecture. The contributions of this paper are twofold: (i) we formalize the adaptive and dynamic VNF allocation problem for mobile networks, and (ii) we propose an approximation algorithm, named A^2VF (*Adaptive Allocation of Virtual Functions*), that ensures efficient on–boarding of network services in polynomial time. Extensive numerical results show that the proposed algorithm runs one order of magnitude faster than the ILP–based placement algorithm, while providing comparable performance in terms of embedding cost.

The rest of this paper is structured as follows. In Sec. II we discuss the related work. The adaptive virtual network function allocation problem is formulated in Sec. III. The A^2VF approximation algorithm is introduced in Sec. IV. Section V provides the performance evaluation. Finally, Sec. VI draws the conclusions while pointing out the future work.

II. Related Work

Virtual Network Embedding (VNE). Efficient mapping of virtual network requests (VNR) or service function chains (SFC) onto a substrate network is a particular type of virtual network embedding (VNE) problem. The problem is *NP*–hard and has been studied extensively in the literature [2], [4]. There are different categories of VNE problems such as inter–domain/intra–domain VNE, survivable VNE, static/dynamic VNE and so on [5].

DaVinci [6] proposed an adaptive resource allocation platform with an ultimate goal of efficiently managing virtual network resources by maximizing the overall network performance across multiple virtual networks. A distributed protocol was derived through optimization decomposition, in order to meet (maximize or minimize) various objective functions of different virtual networks sharing the same substrate network resources. The authors of [7] put forward a VNE problem that considers time–varying resource requirements of a VNR. Their design includes an opportunistic resource sharing–based framework, which consists of two components: macro–level (e.g., node–to–node or link–to–path embedding) and micro–level (e.g., bandwidth requirement) embeddings.

A dynamically adaptive virtual resource allocation algorithm is proposed in [8] for a VNE problem. The algorithm aims at maximizing the acceptance rate, and therefore, InPs' revenue by increasing resource allocation efficiency. The latter is achieved by forecasting the usage rate of resources. The basic strategy is that the VNR is divided into several star–shaped VNRs, then the mapping of each of them is formulated and solved by an approximation bottleneck algorithm as a K–supplier problem. To reconstruct whole VN topology, backtracking algorithm is employed with the objective of minimizing the global mapping cost.

The authors of [9] present a dynamic VNE algorithm, assuming that a substrate node can be reused to embed multiple virtual nodes from the same VNR as long as it has enough capacity. The dynamism of a VNR is considered from the following events perspective: adding/removing a node from the VNR, and increasing/decreasing a virtual node/link requirement. For an increased link requirement, if the substrate link capacity is not enough, path splitting approach is employed.

While many prior works are focused on finding efficient methods to maximize revenue generated for InPs, the work presented in [10], considers optimization of energy consumption, while at the same time, guaranteeing high revenues to the InP. An ILP–based and a heuristic algorithm are proposed to solve the VNE problem for VNRs with dynamically changing demands, which are modelled using the Gaussian distribution.

Notation	Description
N_{nfvi}	Set of substrate nodes
E_{nfvi}	Set of substrate links
$\omega_c^s(n)$	Available CPU resources at node $n \in N_{nfvi}$
$\omega_m^s(n)$	Available memory resources at node $n \in N_{nfvi}$
$\omega_s^s(n)$	Available storage resources at node $n \in N_{nfvi}$
$\omega_r^s(n)$	Available radio resources at node $n \in N_{nfvi}$
$\omega_e^s(e^{nm})$	Available bandwidth resources of link $e^{nm} \in E_{nfvi}$
$\Lambda_n^{c,m,s,r}$	Cost of each unit of node resources
Λ_e	Cost of each unit of link resources

TABLE I: Substrate network notations

Virtual Network Function (VNF) Placement. The VNF placement problem is conceptually similar to component placement in data–centers and clouds. The amount of literature in this domain is thus humbling [11]–[14]. A survey on resource management in cloud computing environments can be found in [3].

In [11], the authors study the problem of placing virtual machine instances on physical containers in such a way to reduce communication overhead and latency. In [12], the authors propose a novel design for a scalable and hierarchical application components placement for cloud resource allocation. The proposed solution operates in a distributed fashion, ensuring scalability, while ensuring performances very close to that of a centralized algorithm. This work is extended in [13] where several algorithms for efficient data management of component–based applications in cloud environments are proposed.

In [14], the elasticity overhead and the trade–off between bandwidth and host resource consumption are jointly considered by the authors who formulate the VNF placement problem. In [15] and [2], a joint node and link mapping algorithm is proposed. While the authors of [1], [16], [17] tackle the problem of dynamic VNF placement. A VNF placement problem is proposed in [18] for radio access networks. In [19], an online VNF scheduling and mapping problem is formulated. The authors propose three greedy algorithms and a tabu search-based heuristic. These algorithms are compared using criteria such as cost, revenue and service processing time and do not consider link constraints, bandwidth requirements and the associated transmission delay between VNFs.

III. Problem Formulation

This paper adopts and extends the network model presented in [20]. Note that as opposed to our work, the authors of [20] do not consider dynamic and adaptive allocation. The objective is to allocate VNFs to physical nodes in s substrate networks so as to optimize the global cost. Each physical node has four types of resources: CPU, memory, storage, and radio, while each physical link has the bandwidth resource. A summary of the notation used in this paper is provided in Table I.

This model considers that the users are able to request service function chains (SFCs) as a directed and acyclic graph

Notations	Description
N_{sfc}	Set of nodes in SFC (VNFs)
E_{sfc}	Set of virtual links
$\omega_c^v(n)$	Requested CPU resources at node $n \in N_{sfc}$
$\omega_m^v(n)$	Requested memory resources at node $n \in N_{sfc}$
$\omega_s^v(n)$	Requested storage resources at node $n \in N_{sfc}$
$\omega_r^v(n)$	Requested radio resources at node $n \in N_{sfc}$
$\omega_b^v(n)$	Requested bandwidth resources at node $n \in N_{sfc}$
$\Omega_b^v(n)$	Reference bandwidth at node $n \in N_{sfc}$
$\omega_e^v(e^{nm})$	Available bandwidth resources of link $e^{nm} \in E_{sfc}$

TABLE II: SFC request notations

$G_{sfc} = (N_{sfc}, E_{sfc})$ in which N_{sfc} is the set of virtual network functions (VNFs) and E_{sfc} is the set of virtual links. A VNF in an SFC requests resources at substrate node n with amounts $\left(\omega_c^v(n), \omega_m^v(n), \omega_s^v(n), \omega_r^v(n), \omega_b^v(n)\right)$ while a virtual link requests bandwidth resource at substrate link e of amount $\omega_e^v(e)$. A reference bandwidth $\Omega_b^v(n)$ was introduced to share radio resources between users equally. A request of radio resources can be in the terms of a fraction of available radio resources (ω_r^v) or an amount of bandwidth ω_b^v. The actual aggregate throughput of the virtual radio node n is denoted as $b(n)$, then the effective bandwidth for virtual radio node $\tilde{\omega}_b^v(n)$ is determined as $\tilde{\omega}_b^v(n) = \begin{cases} \omega_b^v(n) & \text{if } b(n) \geq \Omega_b^v(n) \\ \omega_b^v(n)\frac{b(n)}{\Omega_b^v(n)} & \text{if } b(n) < \Omega_b^v(n) \end{cases}$. A great $\Omega_b^v(n)$ will prefer virtual nodes with good aggregate throughput. Conversely, a small $\Omega_b^v(n)$ tends to treat virtual nodes fairly. The fraction of radio resource can be retrieved from the effective bandwidth by $\omega_r^v(n) = \frac{\tilde{\omega}_b^v(n)}{b(n)}$. Then, the constraint of radio resource of a radio processing node can be expressed as follows.

$$\sum_{n' \in N_{sfc}^b} \frac{\omega_b^v(n')}{\Omega_b^v(n')} \Phi_n^{n'} + \sum_{n' \in N_{sfc}^r} \omega_r^v(n') \Phi_n^{n'} \leq 1, \forall n \in N_{nfvi} \tag{1}$$

More examples and explanations related to $\Omega_b^v(n)$ could be found in [20]. Table II summarizes the notations related to SFC requests.

We consider a time-slotted model, where the system is assumed to be unchanged for the duration of one slot. The new SFC request may arrive in the middle of the slot and will be assigned to the substrate nodes in the next slot. The assignment of VNFs is computed at the beginning of every slot and we can assume that the time-slot is much longer than the migration process [21].

Binary variables $\Phi_n^{n'}$ and $\Phi_e^{e'}$ are introduced to denote the mapping of virtual network functions and virtual edges to substrate nodes and links respectively. $\Phi_n^{n'}$ is 1 when virtual network function n' is hosted by substrate node n and $\Phi_e^{e'}$ is 1 when link e conveys traffic on virtual edge e'. Based on the proposed model in [20], we have following constraints.

- The amount of allocated resources should be less than or equal to the amount of available resources

$$\sum_{n'} \omega_x^v(n') \Phi_n^{n'} \leq \omega_x^s(n), \forall n \in N_{nfvi} \tag{2}$$

, where x could be c,s,m,r corresponding to CPU, storage, memory, and radio resources respectively.

$$\sum_{n'} \omega_e^v(n') \Phi_e^{e'} \leq \omega_e^s(e), \forall e \in E_{nfvi} \tag{3}$$

- Every VNF is mapped to different substrate nodes

$$\sum_{n'} \Phi_n^{n'} \leq 1, \forall n \in N_{nfvi} \tag{4}$$

- Every VNF is mapped at most once

$$\sum_{n} \Phi_n^{n'} \leq 1, \forall n' \in N_{sfc} \tag{5}$$

The above equation means that the model allows one or some VNFs are not mapped when the resources are not sufficient. Note that the solution if any is the maximum number of allocated SFCs by introducing the penalty function (eq. 13) which will be discussed in following paragraphs. The equality constraint could be used, however it might lead to no feasible solution even there is only one VNF cannot be mapped.

- The following constraints enforce a continuous path assigned between substrate nodes n, m hosting VNFs n', m':

$$\sum_{m}^{m>n} \Phi_{e^{nm}}^{e^{n'm'}} - \sum_{m}^{m<n} \Phi_{e^{mn}}^{e^{n'm'}} = \Phi_n^{n'} - \Phi_n^{m'}, \\ \forall i \in N_{nfvi}, \forall e^{n'm'} \in E_{sfc} \tag{6}$$

Unlike the work presented in [20], the cost of placing a VNF includes the cost of resource usage, the transmission cost, and the migration cost. The cost of resource usage is computed as a function of the resources used by a VNF. The transmission cost of an SFC ζ, $x_{d,\zeta}(t)$, is a function of the number of hops between adjacent VNFs. The migration cost of SFC ζ, $x_{m,\zeta}(t)$, is a function of the difference in number of hops between the location in time-slot t and $t+1$ of VNFs. The transmission and migration cost functions can be modeled using the constant-plus-exponential function proposed in [21]:

$$c_{m,\zeta}(t) = \begin{cases} 0, & x_{m,\zeta}(t) = 0 \\ \beta_c + \beta_l \mu^{x_{m,\zeta}(t)} & x_{m,\zeta}(t) > 0 \end{cases} \tag{7}$$

$$c_{d,\zeta}(t) = \begin{cases} 0, & x_{d,\zeta}(t) = 0 \\ \delta_c + \delta_l \theta^{x_{d,\zeta}(t)} & x_{d,\zeta}(t) > 0 \end{cases} \tag{8}$$

Let $d_{i,j}$ be the shortest distance in number of hops between substrate node i and j. Therefore, we have

$$x_{m,\zeta}(t) = \sum_{n' \in N'_\zeta} \sum_{n_1 \neq n_2} d_{n_1,n_2} \Phi_{n_1}^{n'}(t-1) \Phi_{n_2}^{n'}(t) \tag{9}$$

$$x_{d,\zeta}(t) = \sum_{e_{n,m}} \sum_{e_{n',m'} \in \zeta'} d_{n,m} \Phi_{e_{n,m}}^{e_{n',m'}} \tag{10}$$

All costs and distance notations are listed in III. Fig. 1 describes an example of SFC allocation studied in this paper.

Notations	Description
$x_{d,\zeta}(t)$	Transmission distance in number of hops
$x_{m,\zeta}(t)$	Migration distance in number of hops
β_c, β_l, μ	Real-valued parameters of migration cost function
$\delta_c, \delta_l, \theta$	Real-valued parameters of transmission cost function
$d_{n,m}$	distance in number of hops between two substrate nodes n and m

TABLE III: Cost and distance notations

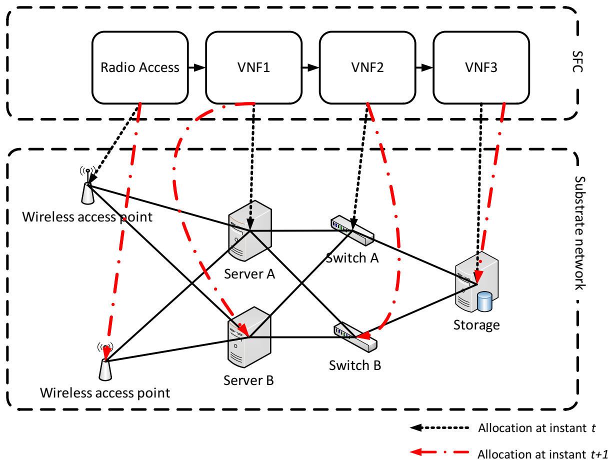


Fig. 1: SFC allocation example

Note that there can be any number of VNFs. In this example, used for illustration, a linear SFC request comprises a radio access node and 3 VNFs. In time-slot t, the radio access node, VNF1, VNF2, and VNF3 are mapped into wireless access point 1, server A, switch A, and the storage server respectively. At the beginning of time-slot $t+1$, the radio access resource, VNF1, and VNF2 are hosted at wireless access point 2, server B, switch B respectively. The migration cost in time-slot $t+1$ is 6 (radio access resource, VNF1, and VNF2 migrate 2 hops each) while the transmission cost is 3 for both time-slot t and $t+1$. The end-to-end hop distances in time-slot t and $t+1$ are similar, but the migration cost is not zero.

Let $\Omega^v_{n,n'} = \omega^v_c(n')\Lambda^c_n + \omega^v_m(n')\Lambda^m_n + \omega^v_s(n')\Lambda^s_n + \omega^v_r(n')\Lambda^r_n$ and $\Omega^v_{e,e'} = \omega^v_e(e')\Lambda_e$.

SFC ζ is not considered assigned when it has at least an unassigned VNF. With the given K SFCs indexed with $1, ..., K$, we introduce a penalty function to maximize the number of admitted SFCs as follows

$$M_\zeta = \begin{cases} (K+1)\, M_0 & \exists \left(n', \zeta\right) : \sum\limits_n \Phi^{n'}_n = 0 \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

where M_0 is the upper-bound cost of allocated SFCs.

Let us denote: $h_\zeta = u\left(\sum\limits_{n' \in N'_\zeta} \sum\limits_n \Phi^{n'}_n - |N'_\zeta|\right)$. $h_\zeta = 1$ when all VNFs of ζ are allocated. Then, we have:

$$\sum_{n' \in N'_\zeta} \sum_n \Phi^{n'}_n \geq |N'_\zeta| h_\zeta. \tag{12}$$

Then, the penalty function could be rewritten as follow

$$M_\zeta = (K+1)\, M_0 \left(1 - h_\zeta\right) \tag{13}$$

The total cost in time-slot t is

$$\sum_\zeta \Bigg(\sum_n \sum_{n'} \Omega^v_{n,n'} \Phi^{n'}_n(t) + \sum_e \sum_{e'} \Omega^v_{e,e'} \Phi^{e'}_e(t) + \\ + c_{m,\zeta}(t) + c_{d,\zeta}(t) + M_0 \left(K+1\right) \left(1 - h_\zeta\right) \Bigg) \tag{14}$$

Lemma 1. *The optimal solution, if exists, is the set of SFCs which has the maximum cardinality.*

Proof. Without loss of generality, let us assume that we have N SFCs. The optimal solution comprises P allocated SFCs indexed $1, ..., P$ and $N-P$ unallocated SFCs indexed $P+1, ..., N$. The cost of each allocated SFC is a_i. We have the total optimal cost as $A^* = a_1 + ... + a_P + M_0\,(N+1)\,(N-P)$.

We assume that there is another feasible solution, with the total cost A', which can admit more SFCs than the optimal solution. Let us call the number of unallocated SFCs in that solution as L and $L \leq N - P - 1$. Without loss of generality, we assume that L unallocated SFCs are from $N-L+1$-th SFC to N-th SFC. The total cost A' is $a'_1 + ... + a'_P + a'_{P+1} + ... + a'_{N-L} + M_0\,(N+1)\,L$.

We have $A^* \leq A'$ since A^* is the cost of the optimal solution. Then, we have $a_1 + ... + a_P + M_0(N+1)(N-P) \leq a'_1 + ... + a'_P + ... + a'_{N-L} + M_0(N+1)L$ which can be rewritten as

$$\left(a_1 - a'_1\right) + ... + \left(a_P - a'_P\right) + M_0(N+1)(N-P) \leq \\ a'_{P+1} + ... + a'_{N-L} + M_0(N+1)L \tag{15}$$

Since a_i and a'_i are less than or equal to M_0, the L.H.S of (15) is greater than or equal to $M_0(N+1)(N-P) - PM_0 = M_0\left[(N+1)(N-P) - P\right]$. Meanwhile, the R.H.S of (15) is less than or equal to $(N-L-P)M_0 + M_0(N+1)L = M_0\left[N(L+1) - P\right]$. Due to $N - P \geq L+1$, L.H.S is greater than or equal to $M_0\left[(N+1)(L+1) - P\right]$. Consequently, R.H.S is less than L.H.S or $A^* \geq A'$. It contradicts to the initial assumption. $\square$

As the distance is an integer, $c_m(x_m(t))$ and $c_d(x_d(t))$ are non-decreasing step functions.

$$c_{m,\zeta}(t) = \beta_c u(x_{m,\zeta}(t)) + \sum_{k=1}^{\infty} u(x_{m,\zeta}(t) - k)\beta_l \left(\mu^k - \mu^{k-1}\right) \tag{16}$$

$$c_{d,\zeta}(t) = \delta_c u(x_{d,\zeta}(t)) + \sum_{k=1}^{\infty} u(x_{d,\zeta}(t) - k)\delta_l \left(\theta^k - \theta^{k-1}\right) \tag{17}$$

Let us denote $\eta^k_{m,\zeta}(t) = u(x_{m,\zeta}(t) - k)$ and $\eta^k_{d,\zeta}(t) = u(x_{d,\zeta}(t) - k)$, so $\eta^k_{m,\zeta}(t)$ and $\eta^k_{d,\zeta}(t)$ are binary variables.

Lemma 2. *$\eta^k_{m,\zeta}(t)$ and $\eta^k_{d,\zeta}(t)$ satisfy the following characteristics*

$$k\eta^k_{m,\zeta}(t) \leq x_{m,\zeta}(t) < k + (X_{m,\zeta}(t) - k)\eta^k_{m,\zeta}(t) \qquad (18)$$

$$k\eta^k_{d,\zeta}(t) \leq x_d(t) < k + (X_{d,\zeta}(t) - k)\eta^k_{d,\zeta}(t) \qquad (19)$$

where $X_{m,\zeta}$ and $X_{d,\zeta}$ are the upper-bound of $x_{m,\zeta}(t)$ and $x_{d,\zeta}(t)$.

Proof. (18) and (19) are similar. We are going to prove the (18). The proof of (19) is similar.

- When $x_{m,\zeta}(t) \geq k$, we have $\eta^k_{m,\zeta}(t) = 1$ following the definition of $\eta^k_{m,\zeta}(t)$ and inequality (18) is satisfied. The case $x_{m,\zeta}(t) < k$ can be similarly proven.
- When $\eta^k_{m,\zeta}(t) = 1$: from inequality (18), we have $k \leq x_{m,\zeta}(t) < X_{m,\zeta}(t)$ that satisfies the definition of $\eta^k_{m,\zeta}(t)$. The case $\eta^k_{m,\zeta}(t) = 0$ can be similarly proven.

□

Lemma 3. *The feasible solution has the elements satisfy $\eta^k_{m,\zeta}(t) \geq \eta^{k+1}_{m,\zeta}(t)$*

Proof.

- When $\eta^k_{m,\zeta}(t) = 0$, we have $u\left(x_{m,\zeta}(t) - k\right) = 0$ which means $0 \leq x_{m,\zeta}(t) < k$. Therefore, $\eta^{k+1}_{m,\zeta}(t) = 0$ and $\eta^k_{m,\zeta}(t) \geq \eta^{k+1}_{m,\zeta}(t)$
- When $\eta^k_{m,\zeta}(t) = 1$, we have $\eta^k_{m,\zeta}(t) \geq \eta^{k+1}_{m,\zeta}(t)$ because $\eta^{k+1}_{m,\zeta}(t) \in \{0,1\}$

□

The objective is to minimize the total cost in (14) under aforementioned constraints. The optimization problem can be written in the following form

$$\begin{aligned} \min \quad & (14) \\ \text{s.t.} \quad & (2),\ (3),\ (4),\ (5),\ (1),\ (6) \\ & (12),\ (18),\ (19) \\ & \Phi^{n'}_n,\ \Phi^{e'}_e,\ \eta_m,\ \eta_d,\ h \in \{0,1\} \end{aligned}$$

The above problem is NP-hard and hence there is no known polynomial time algorithm to solve it. Thus, we propose an approximation algorithm in the next section that takes polynomial time.

IV. Approximation algorithm

The proposed algorithm, Adaptive Allocation VNFs (A^{2}VF) is divided into two sub-algorithms Migration and transmission distance approximation (MEDA) and Node and Link Assignment (NOLA).

The pseudo-code of algorithm A^{2}VF is described in Alg. 1. A^{2}VF begins by solving a relaxed problem in which all integral constraints are removed (e.g. $0 \leq \Phi^{n'}_n \leq 1$). By doing that, we have the fractional solution of the optimization problem $\tilde{\mathcal{S}} = \left(\tilde{\eta}_{m,\zeta}, \tilde{\eta}_{d,\zeta}, \tilde{\Phi}^{n'}_n, \tilde{\Phi}^{e'}_e, \tilde{h}\right)$. The cost of each SFC is estimated based on $\tilde{\mathcal{S}}$. From line 6 to 12, A^{2}VF attempts to round variable h of each SFC to 1 and finds a corresponding feasible solution by executing the sub-algorithms. If there is no feasible solution, it rounds the variable h to 0 and moves to the next SFC. This process lasts until all SFCs are checked.

To find a feasible solution in the aforementioned process, A^{2}VF will call MEDA sub-algorithm presented in Alg. 2. MEDA starts with finding the least fractional entry $\tilde{\eta}_{k_0}$ of the input (line 5). We define distance bound (DB) parameter as the maximum number of physical hops per virtual link. The higher DB means the longer connections between nodes hosting VNFs in the SFC. If the input is η_d, MEDA will compare k_0 with the maximum number of hops of the SFC DB$|E'_\zeta|$. If k_0 is greater than that threshold, k_0 is set to DB$|E'_\zeta| + 1$. Then, MEDA rounds η_{k_0} to 0 by adding a constraint (line 8). All entries $k > k_0$ will be 0 following lemma 3. Line 9 to line 19 are executed if the current solution is feasible. If the input is $\tilde{\eta}_m$, MEDA will execute the pseudo-code from line 10 to line 14. Otherwise, line 15 to line 19 will be executed.

- If $\tilde{\eta}_m$ is the input, MEDA will be called again with input $\tilde{\eta}_d$.
- If $\tilde{\eta}_d$ is the input, NOLA will be called (Alg. 3).

The output of NOLA and MEDA will be (state, solution). The solution will be NULL if the output state is **FAIL**. Otherwise, the solution is a feasible solution. If the output is **SUCCESS**, the next least fractional entry will be considered to find a better η_m (η_d). If the output is **FAIL**and $k_0 >$ DB$|E'_\zeta|$, MEDA is unable to extend the range of η_d and breaks the loop (line 21). Otherwise, MEDA extends the range of η to find a feasible solution (line 23 and 24). At the end of MEDA (when all η_m or η_d are integer), it returns **SUCCESS** if a feasible solution is found.

NOLA comprises two stages: node assignment (line 5 to line 17) and link assignment (line 18 to line 34). In the first stage, each VNF of an SFC will be assigned to a substrate node based on its relaxed solution. Note that $\mathcal{P}$ has been given as input to NOLA from line 15 of Alg. 2 which has a feasible solution. If all $\Phi^{n'}_n{}^0$ are integers, then the list of assigned nodes v^* is extracted from the solution of $\mathcal{P}$. Otherwise, the most fractional entry will be rounded to 1 by adding a constraint at line 11. In case there is no feasible solution, that variable will be rounded to 0. Otherwise, that assignment will be stored in v^* (line 10 to line 17). Note that rounding $\Phi^{n'}_n$ to 0 will create a feasible solution for $\mathcal{P}$ (line 14).

The second stage begins by checking variables $\left\{\Phi^{e'}_e{}^0\right\}$. When there are fractional entries in $\left\{\Phi^{e'}_e{}^0\right\}$, line 19 to line 26 will be executed. From line 27 to line 34, NOLA finds and confirms if it could form a path from the links determined in the previous steps. At the end of NOLA, it will check if all virtual links have been assigned to substrate links and the current cost is less than the initial cost and return the solution with flags.

V. Numerical Results

In this section, the performance of A^{2}VF and ILP-based approaches are compared to highlight the pros and cons of A^{2}VF . Two objective functions are formulated as an ILP problem: with migration and end-to-end distance costs and without them - ILP-ND (no distance). ILPs are solved by

Algorithm 1: Adaptive Allocation VNFs (A^{2}VF)

```
Input: A$^2$VF optimization problem $\mathcal{P}$
Output: Sub-optimal solution $\mathcal{S} = \left(\Phi_n^*, \Phi_e^*, \eta_m^*, \eta_d^*, h_\zeta^*\right)$
Solve the relaxed problem of
  $\mathcal{P} \rightarrow \tilde{\mathcal{S}} = \left(\tilde{\Phi}_n, \tilde{\Phi}_e, \tilde{\eta}_m^*, \tilde{\eta}_d, \tilde{h}_\zeta\right)$;
Sort SFCs in ascending order of their total costs;
Initialize the total cost $z = \infty$;
foreach SFC $\zeta$ do
    Add constraint $h_\zeta = 1$ to $\mathcal{P}$;
    if the relaxed of $\mathcal{P}$ has a feasible solution then
        if MEDA($\tilde{\eta}_m, \mathcal{P}, \zeta, z$) == SUCCESS then
            Feasible solution of $\mathcal{P} \rightarrow \mathcal{S}$;
            z=COST($\mathcal{S}$);
            continue;
    Substitute constraint $h_\zeta = 1$ by $h_\zeta = 0$;
    Solve the relaxed problem of $\mathcal{P} \rightarrow \tilde{\mathcal{S}}$;
```

Algorithm 2: Migration and E2E distance approximation (MEDA)

```
Input: A fractional solution of $\tilde{\eta}_x$ ($\tilde{\eta}_m$ or $\tilde{\eta}_d$),
  optimization problem $\mathcal{P}$, SFC $\zeta$, current cost $z$
Output: A Feasible solution of SFC $\zeta$
Temporary solution $\mathcal{S} = \emptyset$;
while it exists a fractional entry in $\tilde{\eta}_x$ do
    Find the least fractional entry of $\tilde{\eta}_x \rightarrow \tilde{\eta}_{k_0}$;
    if $\tilde{\eta}_x$ is $\tilde{\eta}_d$ and $k_0 >$ DB $\times |E'_\zeta| + 1$ then
        $k_0 =$ DB $\times |E'_\zeta| + 1$;
    Add constraint $\tilde{\eta}_{k_0} = 0$ to $\mathcal{P}$;
    if $\mathcal{P}$ has a feasible solution then
        if $\tilde{\eta}_x$ is $\tilde{\eta}_m$ then
            if MEDA($\tilde{\eta}_d, \mathcal{P}, \zeta, z$) == SUCCESS then
                Returned solution $\rightarrow \mathcal{S}$;
                z=COST($\mathcal{S}$);
                continue;
        if $\tilde{\eta}_x$ is $\tilde{\eta}_d$ then
            if NOLA($\mathcal{P}, \zeta, z$) == SUCCESS then
                Returned solution $\rightarrow \mathcal{S}$;
                z=COST($\mathcal{S}$);
                continue;
    if $\tilde{\eta}_x$ is $\tilde{\eta}_d$ and $k_0 >$ DB $\times |E'_\zeta|$ then
        break;
    else
        Substitute constraint $\tilde{\eta}_{k_0} = 0$ by $\tilde{\eta}_{k_0} = 1$;
        Solve relax $\mathcal{P} \rightarrow \tilde{\eta}_x$;
if $\mathcal{S} \neq \emptyset$ then
    return (SUCCESS, $\mathcal{S}$);
else
    return FAIL;
```

Algorithm 3: Node and Link Assignment (NOLA)

```
Input: SFC $\zeta$ and problem $\mathcal{P}$, current total cost $z$
Output: A feasible set $\mathcal{S}$ of $\Phi_n$ and $\Phi_e$
Temporary solution $\mathcal{S} = \emptyset$;
List of assigned node $v^* = \emptyset$;
foreach VNF $n'_0$ in SFC $\zeta$ do
    if $\Phi_n^{n'_0} \in \{0, 1\}, \forall n$ then
        $v^* \leftarrow \mathcal{P}$;
    else
        while $\exists \Phi_n^{n'_0} \notin \{0, 1\}$ do
            Find the most fractional entry $\Phi_{n_0}^{n'^0}$;
            Add constraint $\Phi_{n_0}^{n'^0} = 1 \rightarrow \mathcal{P}$;
            Solve $\mathcal{P} \rightarrow \mathcal{S}$;
            if $\mathcal{S} == \emptyset$ then
                Substitute $\Phi_{n_0}^{n'_0} = 1$ by $\Phi_{n_0}^{n'_0} = 0$;
                Solve $\mathcal{P} \rightarrow \mathcal{S}$;
            else
                $v^* \leftarrow \left(n'_0, n_0\right)$;
foreach virtual link $e'_0$ in $\zeta$ do
    if $\exists \Phi_e^{e'_0} \notin \{0, 1\}$ then
        while $\exists \Phi_e^{e'_0} \notin \{0, 1\}$ do
            Find the most fractional entry $\Phi_{e_0}^{e'^0}$;
            Add constraint $\Phi_{e_0}^{e'^0} = 1 \rightarrow \mathcal{P}$;
            Solve $\mathcal{P} \rightarrow \mathcal{S}$;
            if $\mathcal{S} == \emptyset$ then
                Substitute $\Phi_{e_0}^{e'_0} = 1$ by $\Phi_{e_0}^{e'_0} = 0$;
                Solve $\mathcal{P} \rightarrow \mathcal{S}$;
    Find a non-cyclic path p connects vertices of $e'_0$
      from $\Phi_e^{e'_0}$;
    foreach physical link $e_0$ in p do
        Add constraint $\Phi_{e_0}^{e'_0} = 1$;
    Solve $\mathcal{P} \rightarrow \mathcal{S}$;
    if $\mathcal{S} == \emptyset$ then
        return FAIL;
    else
        continue;
Total cost of $\mathcal{S}$ is $z'$;
if all virtual links has been assigned and COST $(\mathcal{S}) < z$
  then
    return (SUCCESS, $\mathcal{S}$);
else
    return FAIL;
```

Gurobi® which is deemed as the most advanced ILP solver nowadays. Note that ILP solvers are considered as reference for performance comparison of our proposed heuristic algorithm.

In this paper, we consider linear VNF requests in the time window t. A linear VNF request comprises VNFs connected serially. The placement of VNFs in the previous window $t-1$ is assumed to be known. The number of VNFs in each SFC and the resources of substrate nodes are generated arbitrarily. The reference substrate network is k-ary fat-tree ($k = 4, 6, 8$) in which the leaf nodes are WiFi access points (APs). We assume that leaf nodes have only the radio resource and it is initialized as 1 while other nodes do not have radio resources. The computational, memory, and storage for the substrate nodes are set to 100, while the link resource is set to 1. The cost of using each unit of resources is 1.

The number of VNFs in each request is selected randomly in the range $[3, 6]$. The computational, storage, and memory requirements of each SFC request are uniformly distributed in the range $[25, 30]$ while the link requirements are in the range $[55, 60]$. The radio resource request varies from 0.8 to 1.0. We run 30 simulations with different seeds and consider the average values. The default value of distance bound (DB) is set to infinity so as to optimise the migration distance. Referring to [21], the transmission cost function parameters and the migration cost function parameters are selected as $\theta = 0.8$, $\delta_c = 100$, $\delta_l = -100$ and $\beta_c = 200$, $\beta_l = -100$, $\mu = 0.8$. With these parameters, the algorithms favors the low migration distance.

A. Performance

We first consider the number of accepted SFC requests successfully allocated, calculation time, migration distance and transmission distance with different fat-tree size and number of SFCs. The ILP-based approach provides the optimal solution and is considered as a reference.

Fig. 2 shows the percentage of accepted SFCs obtained by A^{2}VF when the number of SFCs is 1 to 6 and the fat-tree size is 4. The acceptance rate of ILP-based approach is 100% for all case. The acceptance rate of A^{2}VF is 100% when the number of SFC is 1. It drops to 90% when the number of SFCs is 6, i.e., 10% less than the optimal solution.

The calculation time of A^{2}VF under various number of SFCs and fat-tree sizes is expressed in Fig. 3. Generally, the calculation time decreases when the number of SFCs or the fat-tree size decreases. The maximum calculation time, 400 seconds, occurs when there are 8 SFCs in 8-ary fat-tree size substrate network.

In the following simulation, the fat-tree size is fixed at 4. Fig. 4 shows a comparison between the total cost of ILP-based approach and A^{2}VF . The gap in total cost between ILP and A^{2}VF is small when the number of SFCs is 1 or 2 and becomes greater when the number of SFCs increases. At 4 SFCs, the total cost of A^{2}VF is about 10% greater than that of ILP.

Fig. 5 shows that A^{2}VF outperforms ILP-based approaches in terms of calculation time. ILP-ND does not consider mi-

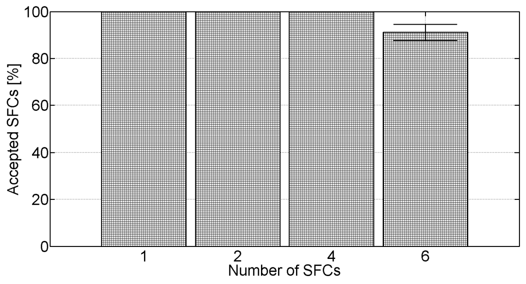


Fig. 2: A^2VF acceptance rate

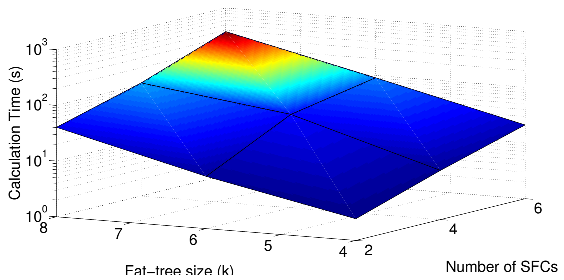


Fig. 3: A^2VF calculation time

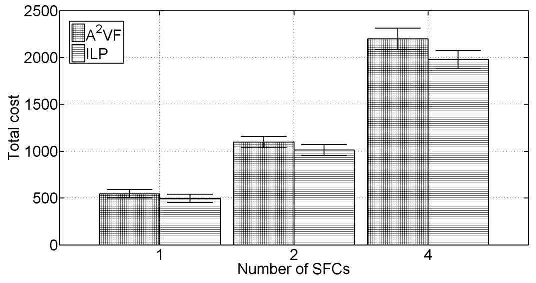


Fig. 4: Average total cost of A^2VF and ILP

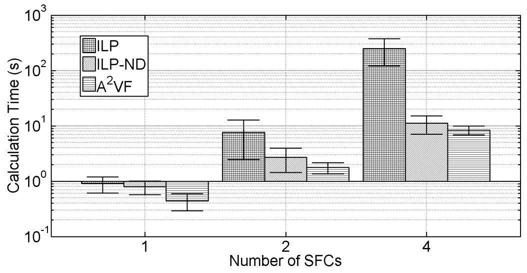
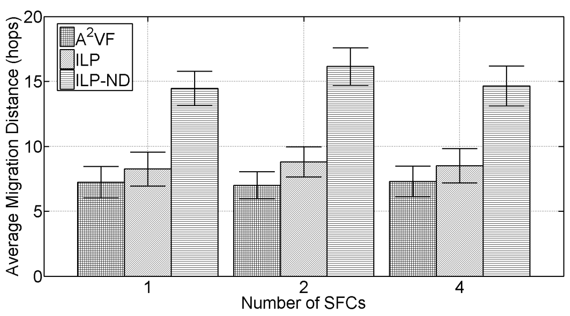
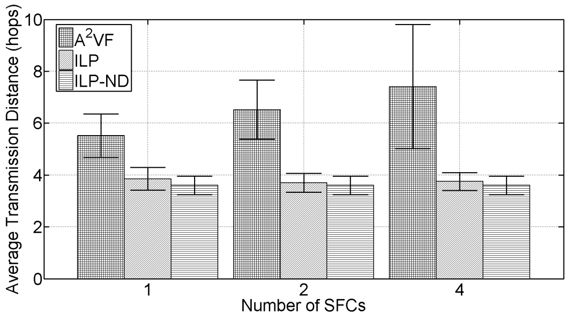


Fig. 5: Calculation time of A^2VF and ILP



(a) Migration distance



(b) Transmission distance

Fig. 6: Average migration and transmission distances of A^2VF, ILP, and ILP-ND

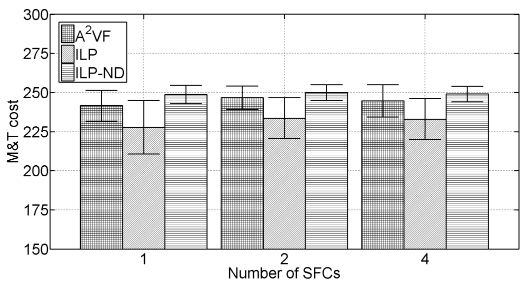


Fig. 7: Migration cost and transmission cost of A^2VF, ILP, and ILP-ND

gration cost and transmission cost, so it is simpler than ILP. Consequently, the calculation of ILP-ND is less than ILP. Generally, A^{2}VF takes less than 10 seconds to find out a feasible solution for 4-ary fat-tree and 1 to 4 SFCs.

Fig 6 shows comparisons of the average migration and transmission distances between A^{2}VF , ILP, and ILP-ND. A^{2}VF with infinite DB has the lowest migration distances while ILP-ND has the greatest migration distance. It is because ILP-ND does not consider migration and transmission costs in its objective function. When it comes to the transmission distances, A^{2}VF has the greatest values while ILP-based approaches have similar values. Although ILP is not the approach that provides lowest migration distance and transmission distance, its migration and transmission combination

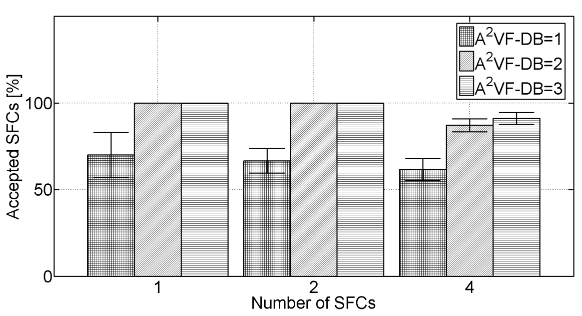


Fig. 8: Average SFC acceptance rate of ILP and A^{2}VF under different distance bounds

cost (M&T) is the lowest as shown in Fig. 7. M&T cost of A^{2}VF is higher than ILP, but it is still less than that of ILP-ND.

B. Distance bound

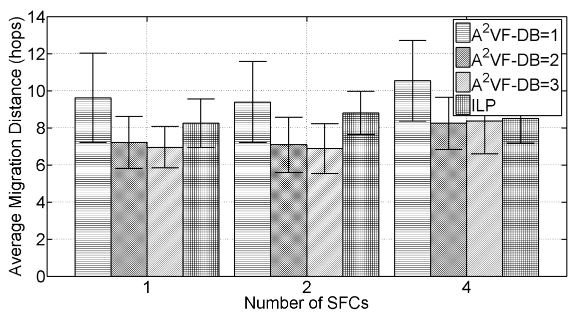
In this section, we analyze the impact of DB on the performance of A^{2}VF . DB is the threshold of the number of physical hops per virtual link. A low DB drives A^{2}VF to provide a lower transmission distance while a high DB means low migration cost. This could help the operators in configuring their networks to meet given requirements. The fat-tree size is 4 and the number of SFCs is from 1 to 4. DB varies from 1 to 3.

The percentage of accepted SFCs is given in Fig. 8. When DB is 1, each virtual link only uses a substrate link. Due to this strictness, the number of accepted SFCs is low. Increasing DB could help to increase the acceptance ratio, especially when the number of SFCs is large. When the number of SFCs is 4, the acceptance ratio is lower than 100% for all values of DB. This is lower than the acceptance ratio of infinity DB shown in Fig. 2 which is 100% when the number of SFCs is 4.

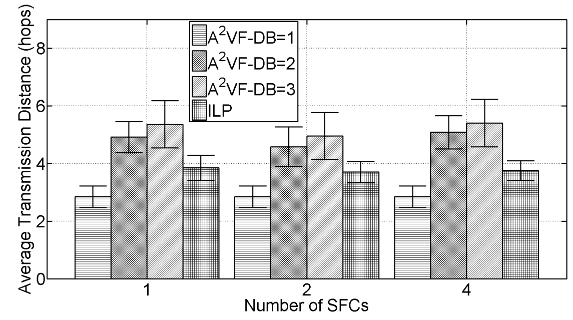
Fig. 9 shows the average migration and transmission distance of ILP and A^{2}VF with DB from 1 to 3. When DB increases, the range of η_d values are extended. Consequently, the transmission distance could be longer in order to have the better migration distance. The total costs are shown in Fig. 10. Since this paper assumes that the migration cost is more expensive than the transmission cost and the high penalty of unallocated SFCs, the lower DB leads to the high total cost. Meanwhile, the total cost of DB $=2$ and DB $=3$ are similar when the number of SFCs is low (one or two SFCs). However, the total cost of DB $=3$ is significantly lower than DB $=2$ when it comes to 4 SFCs. The gap between optimal solution and A^{2}VF could be shortened by setting DB to infinity as shown in Fig 4.

VI. CONCLUSIONS

Network function virtualization is deemed as an essential solution for the next generation networks. In existing literature, the placement of VNFs has been studied intensively. However, the impact of dynamicity and reallocation ability has not been



(a) Migration distance



(b) Transmission distance

Fig. 9: Average migration and transmission distance of ILP and A^2VF under various DB values

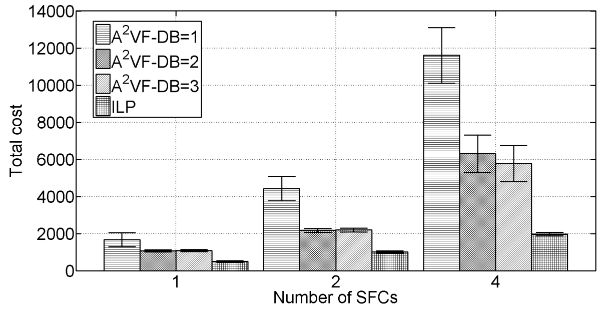


Fig. 10: Average total cost of ILP and A^{2}VF under different distance bounds

addressed. In this paper, the impact of reallocation of SFCs was studied and a model considering dynamic behaviour was provided. To justify the cost of reallocation, the migration distance and transmission distance was considered. The optimisation problem was formulated as an ILP. A heuristic adaptive allocation algorithm A^{2}VF was proposed which finds a near-optimal solution in polynomial time. An extensive evaluation of the heuristic algorithm was provided at the end, with the consideration of different performance metrics such as the migration distance etc. Note that although the nature of A^{2}VF prefers a low migration distance, a threshold of transmission distance (DB) could help to limit the maximum transmission distance. The dynamic adaptiation of DB parameter could be considered in future work.

References

[1] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *Proc. of IEEE CNSM*, Rio, Brasil, 2014.

[2] Z. Despotovic, A. Hecker, A. N. Malik, R. Guerzoni, I. Vaishnavi, R. Trivisonno, and S. A. Beker, "VNetMapper: A fast and scalable approach to virtual networks embedding," in *Proc. of IEEE ICCCN*, Shanghai, China, 2014.

[3] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *J. Netw. Syst. Manage.*, vol. 23, no. 3, pp. 567–619, Jul. 2015.

[4] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *Networking, IEEE/ACM Transactions on*, vol. 20, no. 1, pp. 206 –219, February 2012.

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

[6] J. He, R. Zhang-Shen, Y. Li, C.-Y. Lee, J. Rexford, and M. Chiang, "Davinci: Dynamically adaptive virtual networks for a customized internet," in *Proc. of ACM Proceedings of the 2008 ACM CONEXT Conference*, 2008.

[7] S. Zhang, Z. Qian, J. Wu, S. Lu, and L. Epstein, "Virtual network embedding with opportunistic resource sharing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 816–827, 2014.

[8] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "Adaptive-vne: A flexible resource allocation for virtual network embedding algorithm," in *Proc. of IEEE GLOBECOM*, Anaheim, CA, 2012.

[9] Y. Yuan, C. Wang, C. Wang, B. Zhang, S. Zhu, and N. Zhu, "A Novel Algorithm for Embedding Dynamic Virtual Network Request," in *Proc. of IEEE ICISCE*, Shanghai, 2015.

[10] Z. Zhang, S. Su, J. Zhang, K. Shuang, and P. Xu, "Energy aware virtual network embedding with dynamic demands: Online and offline," *Computer Networks*, vol. 93, pp. 448–459, 2015.

[11] D. Breitgand, A. Epstein, A. Glikson, A. Israel, and D. Raz, "Network aware virtual machine and image placement in a cloud," in *Proc. of IEEE CNSM*, Zurich, Switzerland, 2013.

[12] M. Barshan, H. Moens, and F. De Turck, "Design and evaluation of a scalable hierarchical application component placement algorithm for cloud resource allocation," in *Proc. of IEEE CNSM*, Rio, Brasil, 2014.

[13] M. Barshan, H. Moens, S. Latre, and F. De Turck, "Algorithms for efficient data management of component-based applications in cloud environments," in *Proc. of IEEE NOMS*, Krakow, Poland, 2014.

[14] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic Virtual Network Function Placement," in *Proc. of IEEE CloudNet*, Niagara Falls, Canada, 2014.

[15] R. Guerzoni, R. Trivisonno, I. Vaishnavi, Z. Despotovic, A. Hecker, S. Beker, and D. Soldani, "A novel approach to virtual networks embedding for sdn management and orchestration," in *in Proc. of IEEE NOMS*, Krakow, Poland, 2014.

[16] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The dynamic placement of virtual network functions," in *Proc. of IEEE NOMS*, Krakow, Poland, 2014.

[17] F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *Proc. of IEEE CNSM*, Barcelona, Spain, 2015.

[18] R. Riggio, T. Rasheed, and R. Narayanan, "Virtual network functions orchestration in enterprise WLANs," in *Proc. of IEEE ManFI*, Ottawa, Canada, 2015.

[19] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *in Proc. of IEEE NetSoft*, Seoul, Korea, 2015.

[20] R. Riggio, A. Bradai, D. Harutyunyan, T. Rasheed, and T. Ahmed, "Scheduling wireless virtual networks functions," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 240–252, June 2016.

[21] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. S. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," *CoRR*, vol. abs/1506.05261, 2015. [Online]. Available: http://arxiv.org/abs/1506.05261