Single and Multi-domain Adaptive Allocation Algorithms for VNF Forwarding Graph Embedding

Pham Tran Anh Quang, Abbas Bradai, Kamal Deep Singh, Gauthier Picard, Roberto Riggio

***Abstract*—Network Function Virtualization (NFV) will simplify deployment and management of network and telecommunication services. NFV provides flexibility by virtualizing the network functions and moving them to a virtualization platform. In order to achieve its full potential, NFV is being extended to mobile or wireless networks by considering virtualization of radio functions. A typical network service setup requires the allocation of a Virtual Network Function - Forwarding Graph (VNF-FG). A VNF-FG is allocated considering the resource constraints of the lower infrastructure. 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 also VNF migration. Then we formulate the optimization problem as an Integer Linear Programming (ILP) and provide a heuristic algorithm for allocating multiple VNF-FGs. The idea is that VNF-FGs can be reallocated dynamically to obtain the optimal solution over time. First, a centralized optimization approach is proposed to cope with the ILP-resource allocation problem. Next, a decentralized optimization approach is proposed to deal with cooperative multi-operator scenarios. We adopt AD^3, an ADMM-based algorithm, to solve this problem in a distributed way. The results confirm that the proposed algorithms are able to optimize the network utilization, while limiting the number of reallocations of VNFs which could interrupt network services.**

***Index Terms*—Network function virtualization, VNF-FG embedding, Multi-domain orchestration, Approximation algorithm, Distributed optimization**

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) [1][2][3] 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, an InP could either be 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 [4], virtual network embedding [5], and component placement [6], most of these works focus on the problem of mapping an input virtual network request –often in the form of a VNF Forwarding Graph (VNF-FG) – onto a physical virtualized network substrate – often offering computational as well as networking resources. Nevertheless, most of these works either assume that on-boarded VNF-FGs are not changed after they are mapped onto the substrate network or, when VNF-FG remapping is allowed, they do not consider the migration cost.

In this work, we address the adaptive allocation of VNF-FG for realizing network services in mobile and wireless networks. We model the adaptive and dynamic VNF-FGs allocation problem considering also VNF migration. Then we formulate the optimization problem as an Integer Linear Programming (ILP). As the optimization problem is found to be NP-hard, we provide a heuristic algorithm for allocating multiple VNF-FGs, which can provide near-optimal solutions in polynomial time. We expect MVNOs to specify their requests in terms of a VNF-FG. 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 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. Practically, the substrate networks could be comprised of multiple domains which are controlled by different operators. Therefore, we decompose and distribute the VNF-FG allocation problem over a set of operators. Thanks to our proposed distributed scheme, the operators are able to control their resources and take advantages of extending their services through collaboration and parallelizing the optimization process. This paper extends the work presented in [7] by considering multi-domain orchestration as well as dynamic and adaptive allocation. The cost is revised by adding the migration cost and the transmission cost beside deployment costs. The objective is to allocate maximum number of VNF-FGs to physical nodes in substrate networks with an optimal global cost. The contributions of this paper are threefold:

(i) we formalize the adaptive and dynamic VNF-FG allocation problem for wireless networks,
(ii) we propose a centralized approximation algorithm, named A^{2}VF (*Adaptive Allocation of Virtual Functions*),

that ensures efficient embedding of VNF-FG in polynomial time, and

(iii) we propose a decentralized optimization algorithm, named AD^3-VA (*AD^3-based VNF allocation*), that enables cooperation between domains to deploy a VNF-FG, which is the first application and adaptation of the AD^3 algorithm to this very problem, to the best of our knowledge.

Extensive numerical results show that the proposed centralized algorithm, A^{2}VF , runs one order of magnitude faster than the ILP–based placement algorithm, while providing comparable performance in terms of embedding cost. The decentralized approximation algorithm, AD^3-VA, has a comparable performance to A^{2}VF in moderately complex cases.

The rest of this paper is structured as follows. In Section II we discuss the related work. The adaptive VNF-FG embedding problem is formulated in Section III. The A^{2}VF approximation algorithm is introduced in Section IV. Section V presents the decentralized approximation algorithm AD^3-VA. Section VI provides the performance evaluation. Finally, Section VII draws the conclusions while pointing out the future work.

II. Related Work

This section presents the related work and some background. The text below first discusses the VNF-FG embedding in single and multi-domain scenarios. Then, related work and background on distributed optimization are provided.

A. Single Domain VNF-FG Embedding

Many works have been proposed in the literature on VNF-FG embedding in single domain scenarios as in [8], [9], [10], [11], [12], [13]. In [8], authors propose a heuristic for VNF-FG embedding onto the underlying physical network. The main idea of this work is to sort the VNFs according to the ratio of outgoing flows with regards to the incoming flows, and start by embedding the VNFs with high ratio first then the low ratio ones. The problem of VNF-FG embedding is modeled as an ILP problem in [9] and a heuristic is proposed to solve it. The main goal of this work is to minimize the utilisation of physical resources while meeting the QoS requirements of the VNF-FG. The proposed heuristic decomposes the forwarding graph. Then, the VNFs of the decomposed graph are embedded onto the physical network based on a backtracking mechanism.

A concept of hybrid NFV environment is introduced in [10], where a part of a service is provided by dedicated physical nodes and the other is provided by a virtual network. The problem is modeled as an ILP and a heuristic is used to map the virtual part of the network on the physical resources.

Authors in [11] propose to jointly optimize the VNF-FG design and VNF-FG embedding. Their proposed approach exploits the feedback of mapping the VNF-FG to optimize the VNF-FG design. The objective of this work is to optimize the total bandwidth consumption.

While many prior works focused on finding efficient methods to maximize revenue generated for infrastructure providers, the work presented in [12] considers optimization of energy consumption, while at the same time, guaranteeing high revenues for the InP. An ILP–based algorithm and a heuristic algorithm are proposed to solve the VNF-FG embedding problem with dynamically changing demands, which are modeled using the Gaussian distribution. Similarly, in [14], the authors proposed a heuristic algorithm for solving VNF-FG embedding problem aiming to reduce the global energy consumption. The extension of this work [13] considered long-term and short-term QoS constraints besides energy efficiency.

Some works [15], address the problem in two steps i.e. first VNFs are placed and then chaining is used to steer traffic. We use joint optimisation which is better than such two-step solutions, as pointed out and also used in [16]. However, [16] neither considers dynamic embedding nor distributed scenario, which is considered in our paper. For more related work, interested readers are pointed to a survey paper [17].

All the aforementioned algorithms are compared using criteria such as cost, revenue and service processing time and do not consider the dynamic embedding and re-embedding of the VNF-FG. Our work considers this possibility in this paper and also proposes both centralized and decentralized solutions for dynamic VNF-FG allocation. The decentralized solution can be applied to multi-domain VNF-FG embedding which is discussed in the section below.

B. Multi-domain VNF-FG Embedding

The physical resources providers involved in VNF-FG would mostly like to keep the information about their resources confidential, which in turn motivates our distributed solution for multi-domain VNF-FG allocation.

The NFV reference architecture MANO, proposed by ETSI [18], does not provide any specifications for VNF-FG allocation over administratively different infrastructure domains. A few recent works address VNF orchestration in multi-domain. Munoz et al. [19] addressed the connectivity of VNF-FG over multi-technological domains in SDN-based architecture. However, the proposed integrated SDN/NFV orchestration architecture is based on a centralized orchestrator, which does not fit the case with multi-operators. CloudNFV [20] proposed to orchestrate VNF-FG in a distributed cloud architecture. Their work differs from our contribution in the fact that even if the proposed architecture is distributed over distributed clouds, it is still managed by a single orchestrator which belongs to the same provider. Other works employ a distributed orchestrator which replicates a large scale global view. However such concept has some disadvantages such as the global view can be inconsistent, complexity is high and there is the deficiency of confidentiality [21], [22]. Path Computation Engine (PCE) based solutions are proposed in [21] for mapping virtual optical network in multi-domain networks. The main drawback of this solution is the fact that the parent PCE is obliged to compute all inter-domain paths and there is high signaling overhead.

The X-MANO project [23] proposes a new framework towards inter-domain orchestration. Authors of this work proposed a set of interfaces and abstractions to coordinate the inter-domain orchestration process (including on-boarding,

scaling, and termination) while preserving the confidentiality of each domain. A proof of concept of X-MANO is provided in [24]. However, in this project authors did not propose any mechanism or approach for the inter-domain orchestration, they proposed a general framework which can be used by the community to implement their own mechanisms. Moreover, the 5-GEx [25] European project proposed a new architecture which aims to enable cross-domain orchestration of services over multiple administrations or over multi-domain single administrations. The objective of this architecture is to enable end-to-end network and service elements to mix in multi-vendor, heterogeneous technology and resource environments. This project proposes a new architecture. More information about the 5-GEx architecture are provided in the beginning of Section V.

In this work, we consider the 5-GEx architecture as an example architecture where our proposal can be applied. We note that the solution which we propose is generic and can be adopted in any distributed NFV orchestration architecture.

C. Distributed Cooperative Algorithms and AD^3

Allocating VNF-FGs, composed of several network functions, traversing several domains or operators can be viewed as a decomposition and a distribution of an allocation problem. Thus, the multiple stakeholders manage their own part of the problem and cooperate for the collective assignment.

Alternating Direction Method of Multipliers (ADMM) [26] is a well-known method to decompose and distribute a problem. ADMM has been adopted to solve several networking problems such as resource allocation [27]. However, ADMM was designed for continuous variables and cannot be used directly for problems which have integer variables. In [28], the discrete functions are approximated by polynomial continuous functions; nevertheless, the quality of solution depends on the accuracy of the approximation. The major advantage of ADMM is adaptability in various large-scale distributed problems. However, the slow convergence of ADMM prevents its application to complex problems such as resource allocation for VNF-FG embedding in wireless networks.

Recently, a new ADMM-based algorithm, the so-called AD^3 (Alternating Directions Dual Decomposition) has been proposed in the realm of the machine learning literature [29]. AD^3 has extra interesting features as compared to other message-passing algorithms in the machine learning literature: it reaches consensus faster than other algorithms [30], [31], and it neither has the convergence problems of Max-Product Linear Programming (MPLP) [32] nor the instability problems of Norm-Product Belief Propagation [33]. This unveils the possibility of employing AD^3 to approximate constrained optimization problems. To the best of our knowledge, our paper is the first study on the application of AD^3 to dynamic resource allocation in wireless networks.

III. VNF-FG Allocation Problem

A. VNF-FG Placement and Chaining

Fig. 1 describes the VNF-FG use case that we target in this paper. The use case is detailed in [34], where one or

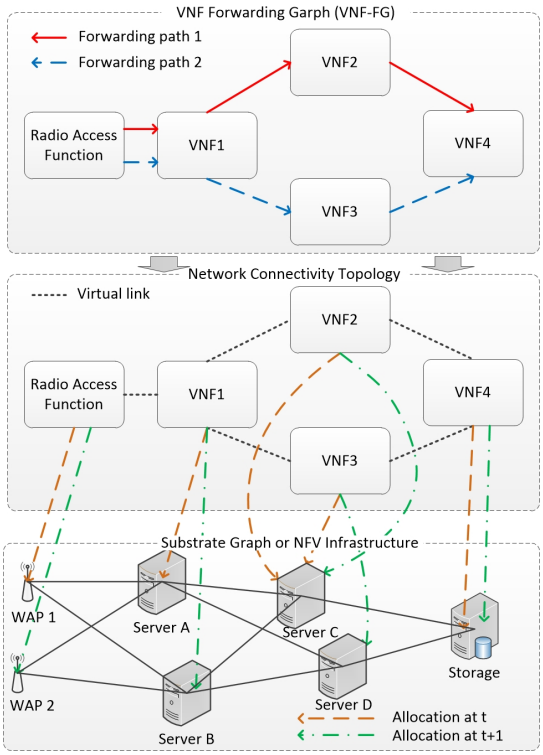


Fig. 1: VNF-FG example

multiple VNF forwarding graphs can be used for implementing a network service. A VNF connects to another VNF using a virtual link (VL) through VNF interfaces called connection points (CP). A VL may be composed of one or several physical links. A forwarding path (FP) is an ordered list of CPs forming a sequence or a chain of VNFs which will be traversed by traffic flows [35]. Finally, a VNF-FG may consist of multiple FPs. Based on the VNF-FG, a network connectivity topology (NCT) is formed using a set of virtual links. It consists of VLs connecting different VNFs as shown in Figure 1. NCT is instantiated over NFV Infrastructure (NFVI) while considering the topological constraints as well as VNF resource requirements. For example, in Fig. 1, there are 2 forwarding paths: red (FP1) and blue (FP2). Assume that FP1 will serve a flow of 3 Mb/s and FP2 will serve a flow of 2 Mb/s. It can be seen in Fig. 1 that both forwarding paths pass through the VL between the radio access function and VNF1. Thus, considering these bandwidth requirements, a 5 Mb/s VL will need to be instantiated in the NCT between radio access function and VNF1. Other VLs such as the one between VNF1 and VNF2 will be instantiated according the their requirements i.e. 3 Mb/s for this particular case.

A VNF-FG deployment system at its core implements the NFV orchestrator and VNF manager defined in the NFV reference architecture framework [36]. They are responsible for orchestration and management of resources, lifecycle management such as instantiation, update, termination, etc. They invoke the modules defined in the NFV reference architecture to deploy VNF-FG. An example of steps for VNF-FG deployment are provided in [35]. First a VNF-FG request arrives. If the request is valid then a template is generated for NCT trans-

Notation	Description
N_s	Set of substrate nodes
E_s	Set of substrate links
$q_c(n)$	Available CPU resources at node $n \in N_s$
$q_m(n)$	Available memory resources at node $n \in N_s$
$q_s(n)$	Available storage resources at node $n \in N_s$
$q_r(n)$	Available radio resources at node $n \in N_s$
$q_e(e^{nm})$	Available bandwidth resources of link $e^{nm} \in E_s$
$\Lambda_n^{c,m,s,r}$	Cost of each unit of node resources
Λ_e	Cost of each unit of link resources

TABLE I: Substrate network notations

Notations	Description
$N_{v,\zeta}$	Set of nodes in VNF-FG ζ
$E_{v,\zeta}$	Set of virtual links in VNF-FG ζ
$\omega_c(n')$	Requested CPU resources of VNF n'
$\omega_m(n')$	Requested memory resources of VNF n'
$\omega_s(n')$	Requested storage resources of VNF n'
$\omega_r(n')$	Requested radio resources of n'
$\omega_b(n')$	Requested bandwidth resources of node n'
$\Omega_b(n')$	Reference bandwidth at node n'
$\omega_e(e^{nm})$	Requested bandwidth of link$e^{nm} \in E_{v,\zeta}$
$N_{v,\zeta}^b$	Set of nodes requesting radio resources in bandwidth
$N_{v,\zeta}^r$	Set of nodes requesting radio resources in a fraction of available radio resource

TABLE II: VNF-FG request notations

lation, placement and chaining. Then a placement and chaining algorithm, which is the topic of this paper, is executed. The algorithm updates the NCT template by specifying where to place VNFs and which virtual links to instantiate. Then the VNFs and the forwarding paths are instantiated. Finally, the requested VNF-FG is ready to be used.

Note that in this paper, we focus on joint VNFs placement and chaining which is better than tackling placement and chaining separately, because topological as well as resource requirements are considered together. Moreover, we focus on dynamically allocating VNF-FG to the substrate graph (or the NFV infrastructure) as different requests arrive over time. By allowing migration of functions, we allocate resources more efficiently and enhance the utilization of the substrate networks, especially when there are multiple VNF-FG demands. Continuing the above example, based on Fig. 1, it can be seen that at time t the VNFs are allocated over NFVI. Now another request arrives. Depending on costs such as that of migration and gains in terms of resource utilization efficiency, it might be optimal to migrate some VNFs to other locations in NFVI. This is for optimization process to find out. In case migration turns out to be optimal, some VNFs are migrated at $t+1$. Migration is elaborated more later in next sub-section.

Note that we consider two scenarios: in the first, we consider a single domain VNF-FG deployment, where both infrastructure and VNF-FG are under control of a single operator. The second scenario is when a VNF-FG is deployed over multiple domains, then a cooperative optimization scheme, based on message passing algorithm, is proposed. These two scenarios are presented in Section IV and Section V, respectively.

B. Problem Formulation

Our system model is described here and a summary of the notation used in this paper is provided in Table I. We assume that VNF-FG requests arrive over time. VNF-FG in turn will be used for realizing network services. Generally, the VNF-FG may compose of multiple forwarding paths. Based on that graph, a network connectivity graph is formed as a directed graph $G_{v,\zeta} = (N_{v,\zeta}, E_{v,\zeta})$ in which $N_{v,\zeta}$ is the set of virtual network functions (VNFs) and $E_{v,\zeta}$ is the set of virtual links of VNF-FG ζ. VNFs are allocated to physical nodes in 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 VNF n' requests resources with amounts $\big(\omega_c(n'), \omega_m(n'), \omega_s(n'), \omega_r(n'), \omega_b(n')\big)$ while a virtual link e' requests bandwidth resource of amount $\omega_e(e')$. Radio resources can be provided either in terms of fraction of available radio resources (ω_r) or in terms of bandwidth (ω_b). Let us denote N_v^r and N_v^b as the nodes request radio resources in terms of fraction of radio resources and bandwidth respectively. In the first case, the users identify the fraction of radio resources allocated at a node. Meanwhile, the users determine the amount of bandwidth assigned to the node in the latter case. A reference bandwidth $\Omega_b(n)$ was introduced to share radio resources between users equally. Table II summarizes the notations related to VNF-FG. A request of radio resources can be in the terms of a fraction of available radio resources (ω_r) or an amount of bandwidth ω_b. The actual aggregate throughput of the virtual radio node n' is denoted as $b(n')$, then the effective bandwidth for virtual radio node $\tilde{\omega}_b(n')$ is $\tilde{\omega}_b(n') = \begin{cases} \omega_b(n') & \text{if } b(n') \geq \Omega_b(n') \\ \omega_b(n')\frac{b(n')}{\Omega_b(n')} & \text{if } b(n') < \Omega_b(n') \end{cases}$. A great $\Omega_b(n)$ will prefer virtual nodes with good aggregate throughput. Conversely, a small $\Omega_b(n)$ tends to treat virtual nodes fairly. The fraction of radio resource can be retrieved from the effective bandwidth by $\omega_r(n') = \frac{\tilde{\omega}_b(n')}{b(n')}$. Then, the constraint of radio resource of a radio processing node can be expressed as follows.

$$\sum_{n' \in N_v^b} \frac{\omega_b(n')}{\Omega_b(n')} \Phi_n^{n'} + \sum_{n' \in N_v^r} \omega_r(n') \Phi_n^{n'} \leq 1, \forall n \in N_s \quad (1)$$

We consider a time-slotted model, where the system is assumed to be unchanged for the duration of one slot. A new VNF-FG request may arrive in the middle of the slot and will be assigned to the substrate nodes in the next slot. Although it could cause additional delay in allocation, an adaptive time-slot, which will be shortened when a new VNF-FG request arrives, can mitigate the delay in resource allocation. This issue is out of scope of this paper and could be considered in future research. 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 [37]. Note that as opposed to our work, the authors of [7] do not consider dynamic and adaptive allocation. They also do not consider migration.

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'. The model proposed in [7] comes with following constraints.

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

$$\sum_{n'} \omega_x(n')\Phi_n^{n'} \leq q_x(n), \forall n \in N_s \tag{2}$$

where x could be c, m, s, or r represented for CPU resources, memory resources, storage resources, and radio resources respectively. For the links, we have

$$\sum_{e'} \omega_e(e')\Phi_e^{e'} \leq q_e(e), \forall e \in E_s \tag{3}$$

- Every VNF is mapped at most once

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

In [7], the left hand side expression of Eq. (4) is equal to 1. This could lead to infeasible status when solving by ILP solver since there may not be enough resources to accommodate all VNF-FGs in networks. To avoid this, we relax the constraints (4). Although Eq. (4) allows $\Phi_n^{n'} = 0, \forall n, n'$, the penalty function presented in Eq. (6) leads to an optimal solution comprising positive values of $\Phi_n^{n'}$.

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

$$\sum_{m} \Phi_{e^{nm}}^{e^{n'm'}} - \sum_{m} \Phi_{e^{mn}}^{e^{n'm'}} = \Phi_n^{n'} - \Phi_n^{m'}, \quad \forall n \in N_s, \forall e^{n'm'} \in E_v \tag{5}$$

Note that Equation 5 is based on multicommodity flow model in which the L.H.S is the difference between the total incoming flows and the total outgoing flows, while the R.H.S is to indicate if a substrate node hosts the origin or destination of the virtual link.

Unlike the work presented in [7], 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 a VNF-FG ζ, $c_{d,\zeta}(t)$, is a function of the total number of hops between adjacent VNFs. It is assumed to be the transmission cost incurred by the users of the network service realized by the VNF-FG. The migration cost of VNF-FG ζ, $c_{m,\zeta}(t)$, is a function of the difference in number of hops between the location in time-slot t and $t+1$ of VNFs. It is 0 if there was no migration. Thus, it is assumed that migrating a VNF incurs cost proportional to the distance between initial and final location. Note that the parameter which is the number of hops can be easily replaced in the model by some other equivalent parameter. All notations related to costs and distances are listed in Table III. The transmission and migration cost functions can be modeled using the constant-plus-exponential function proposed in [37]:

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	Parameters of migration cost function
$\delta_c, \delta_l, \theta$	Parameters of transmission cost function
$d_{n,m}$	distance (hops) between nodes n and m

TABLE III: Cost and distance notations

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

and

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

The migration distance is the relative difference in the amount of total shortest distance (hops) between the locations of VNFs at instance $t-1$ and at instance t. The transmission distance is the total shortest distance (hops) between VNFs in VNF-FG. The transmission distance ($x_{d,\zeta}$) and migration distance ($x_{m,\zeta}$) are integers (number of hops), while the corresponding costs are fractional values. Let $d_{i,j}$ be the shortest distance in number of hops between substrate node i and j. Therefore, we have

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

and

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

For sake of readability, we use Figure 1 as an example of transmission and migration distances. Note that there can be any number of VNFs and forwarding paths. In this example, a VNF-FG comprises a virtual radio access node and 4 VNFs. In time-slot t, the virtual radio access node, VNF1, VNF2, VNF3, and VNF4 are mapped finally into wireless access point (WAP) 1, server A, server C (VNF2 and VNF3), and the storage server respectively. At the beginning of time-slot $t+1$, the virtual radio access node, VNF1, and VNF3 are hosted at WAP 2, server B, server D respectively; while the hosts of VNF2 and VNF4 are unchanged. The migration distance in time-slot $t+1$ is 6 (radio access resource, VNF1, and VNF3 migrate 2 hops each) while the transmission distance is 5 for both time-slot t and $t+1$.

Let $\Omega_{n,n'} = \omega_c(n')\Lambda_n^c + \omega_m(n')\Lambda_n^m + \omega_s(n')\Lambda_n^s + \omega_r(n')\Lambda_n^r$ and $\Omega_{e,e'} = \omega_e(e')\Lambda_e$ as the cost of deploying VNF n' at substrate node n and virtual link e' at substrate link e.

VNF-FG ζ is not considered assigned unless all of its VNFs are assigned. For a given K and VNF-FGs indexed with $1, ..., K$, we introduce a penalty function to maximize the number of admitted VNF-FGs as follows:

$$M_\zeta = \begin{cases} (K+1)\, M_0 & \exists \left(n', \zeta\right) : \sum_n \Phi_n^{n'} = 0 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where M_0 is the upper-bound cost of allocated VNF-FGs.

This penalty impacts the characteristics of the optimal solution. A high penalty means the number of allocated VNF-FGs is favored, while a low penalty means short migration and transmission distances are more preferred. The value of penalty shown in Eq. (6) guarantees the maximum number of allocated VNF-FGs as proven in Lemma 3. Let us denote $h_\zeta = u\left(\sum_{n' \in N_{v,\zeta}} \sum_{n} \Phi_n^{n'} - |N_{v,\zeta}|\right)$, where $u(x)$ is a step function ($u(x) = 1$ when $x \geq 0$ and $u(x) = 0$ when $x < 0$). Note that $h_\zeta = 1$ when all VNFs of ζ are allocated. Then, we have:

$$\sum_{n' \in N_{v,\zeta}} \sum_{n} \Phi_n^{n'} \geq |N_{v,\zeta}| h_\zeta. \tag{7}$$

Then, the penalty function could be rewritten as follow

$$M_\zeta = (K+1) M_0 \left(1 - h_\zeta\right) \tag{8}$$

The total cost at time-slot t is

$$\sum_{\zeta} \Bigg(\sum_{n} \sum_{n'} \Omega_{n,n'} \Phi_n^{n'}(t) + \sum_{e} \sum_{e'} \Omega_{e,e'} \Phi_e^{e'}(t) + c_{m,\zeta}(t) + c_{d,\zeta}(t) + M_0 (K+1) \left(1 - h_\zeta\right) \Bigg) \tag{9}$$

Since the distance is an integer, $c_m(x_m(t))$ and $c_d(x_d(t))$ are non-decreasing step functions. $c_{m,\zeta}(t) = \beta_c u(x_{m,\zeta}(t)) + \sum_{k=1}^{\infty} u(x_{m,\zeta}(t) - k) \beta_l \left(\mu^k - \mu^{k-1}\right)$ and $c_{d,\zeta}(t) = \delta_c u(x_{d,\zeta}(t)) + \sum_{k=1}^{\infty} u(x_{d,\zeta}(t) - k) \delta_l \left(\theta^k - \theta^{k-1}\right)$

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

Lemma 1. *$\eta_{m,\zeta}^k(t)$ and $\eta_{d,\zeta}^k(t)$ satisfy the following characteristics*

$$k\eta_{m,\zeta}^k(t) \leq x_{m,\zeta}(t) < k + (X_{m,\zeta}(t) - k)\eta_{m,\zeta}^k(t) \tag{10}$$

$$k\eta_{d,\zeta}^k(t) \leq x_d(t) < k + (X_{d,\zeta}(t) - k)\eta_{d,\zeta}^k(t) \tag{11}$$

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

Proof. (10) and (11) are similar. We are going to prove the (10). The proof of (11) is similar. If $x_{m,\zeta}(t) \geq k$, we have $\eta_{m,\zeta}^k(t) = 1$ following the definition of $\eta_{m,\zeta}^k(t)$ and inequality (10) is satisfied. The case $x_{m,\zeta}(t) < k$ can be similarly proven. If $\eta_{m,\zeta}^k(t) = 1$: from inequality (10), we have $k \leq x_{m,\zeta}(t) < X_{m,\zeta}(t)$ that satisfies the definition of $\eta_{m,\zeta}^k(t)$. The case $\eta_{m,\zeta}^k(t) = 0$ can be similarly proven. □

The objective is to minimize the total cost in (9) under aforementioned constraints. The optimization problem can be written in the following form:

Problem 1 (Adaptive VNF-FG Allocation)**.**

$$\begin{aligned} \min \quad & (9) \\ \textit{s.t.} \quad & (1),\ (2),\ (3),\ (4),\ (5) \\ & (7),\ (10),\ (11) \\ & \Phi_n^{n'},\ \Phi_e^{e'},\ \eta_m,\ \eta_d,\ h_\zeta \in \{0,1\}, \forall \zeta \end{aligned}$$

This optimization problem has the following characteristics.

Lemma 2. *The feasible solution to the optimization problem has the elements which satisfy $\eta_{m,\zeta}^k(t) \geq \eta_{m,\zeta}^{k+1}(t)$*

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

Lemma 1 and Lemma 2 provide limits of $\eta_{m,\zeta}^k(t)$ and $\eta_{d,\zeta}^k(t)$, thus reducing the searching complexity of the proposed algorithm presented in Section IV.

Lemma 3. *The optimal solution, if exists, is the set of VNF-FGs which has the maximum cardinality.*

Proof. Without loss of generality, let us assume that we have N VNF-FGs. The optimal solution comprises P allocated VNF-FGs indexed $1, ..., P$ and $N - P$ unallocated VNF-FGs indexed $P+1, ..., N$. The cost of each allocated VNF-FG 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 VNF-FGs than the optimal solution. Let us call the number of unallocated VNF-FGs in that solution as L and $L \leq N - P - 1$. Without loss of generality, we assume that L unallocated VNF-FGs are from $N - L + 1$-th VNF-FG to N-th VNF-FG. 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 $\sum_{i=1}^{P} \left(a_i - a'_i\right) + M_0(N+1)(N-P) \leq \sum_{j=P+1}^{N-L} \left(a'_j\right) + M_0(N+1)L$. Since a_i and a'_i are less than or equal to M_0, the L.H.S 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 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 the initial assumption. □

Problem 1 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. Centralized Approximation Algorithm

The proposed algorithm to solve Problem 1, Adaptive Allocation VNFs (A^2VF) is divided into two sub-algorithms: Migration and transmission distance approximation (MEDA) and Node and Link Assignment (NOLA). The pseudo-code of A^2VF is described in Algorithm 1. A^2VF begins by solving the relaxed problem. 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 VNF-FG is estimated based on $\tilde{\mathcal{S}}$. From line 6 to 13, A^2VF attempts to round variable h of each VNF-FG to 1 and finds a corresponding feasible solution by executing

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

1 **Input:** A^{2}VF optimization problem $\mathcal{P}$
2 **Output:** Sub-optimal solution $\mathcal{S} = \left(\Phi_n^*, \Phi_e^*, \eta_m^*, \eta_d^*, h_\zeta^*\right)$
3 Solve relaxed $\mathcal{P} \rightarrow \tilde{\mathcal{S}} = \left(\tilde{\Phi}_n, \tilde{\Phi}_e, \tilde{\eta}_m^*, \tilde{\eta}_d, \tilde{h}_\zeta\right)$;
4 Sort VNF-FGs in ascending order of their total costs;
5 Initialize the total cost $z = \infty$;
6 **foreach** *VNF-FG* ζ **do**
7 Add constraint $h_\zeta = 1$ to $\mathcal{P}$;
8 **if** *the relaxed of* $\mathcal{P}$ *has a feasible solution* **then**
9 **if** MEDA$(\tilde{\eta}_m, \mathcal{P}, \zeta, z)$ == **SUCCESS then**
10 Feasible solution of $\mathcal{P} \rightarrow \mathcal{S}$ and $z = cost\,(\mathcal{S})$;
11 **continue**;
12 Substitute constraint $h_\zeta = 1$ by $h_\zeta = 0$;
13 Solve relaxed $\mathcal{P} \rightarrow \tilde{\mathcal{S}}$;

the sub-algorithms. If there is no feasible solution, it rounds the variable h to 0 and moves to the next VNF-FG. This process lasts until all VNF-FGs are checked.

To find a feasible solution in the aforementioned process, A^{2}VF will call MEDA sub-algorithm presented in Algorithm 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 VNF-FG. If the input is η_d, MEDA will compare k_0 with the maximum number of hops of the VNF-FG DB$|E_{v,\zeta}|$. If k_0 is greater than that threshold, k_0 is set to DB$|E_{v,\zeta}| + 1$. Then, MEDA rounds η_{k_0} to 0 by adding a constraint (line 8). All entries $k > k_0$ will be 0 following lemma 2. Line 10 to line 17 are executed if the current solution is feasible. If the input is $\tilde{\eta}_m$, MEDA will execute the pseudo-code from line 11 to line 13. Otherwise, line 15 to line 17 will be executed. When $\tilde{\eta}_m$ is the input, MEDA will be called again with input $\tilde{\eta}_d$. Otherwise, NOLA will be called (Algorithm 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$|E_{v,\zeta}|$, MEDA is unable to extend the range of η_d. Otherwise, MEDA extends the range of η to find a feasible solution (line 19). At the end of MEDA (when all η_m or η_d are integer), it returns **SUCCESS** if a feasible solution is found. Otherwise, it returns **FAIL**.

NOLA comprises two stages: node assignment (line 5 to line 14) and link assignment (line 15 to line 27). In the first stage, each VNF of a VNF-FG 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 Algorithm 2 which has a feasible solution. If all $\Phi_n^{n'0}$ are integers, 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 10. In case there is no feasible solution, that variable will be rounded to 0. Otherwise, that assignment will be stored in v^* (line 11 to line 14). Note that rounding $\Phi_n^{n'}$ to 0 will create a feasible solution for $\mathcal{P}$ (line 12).

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 17 to line 22 will be executed. From line 23 to line 27, 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.

The relaxed problem is a linear program which can be solved in polynomial time [38]. Let us denote $\mathcal{O}_{RL}$ as the complexity of the relaxed problem. Alg. 3 has the worst case complexity of $\mathcal{O}_{NOLA} = \sum_\zeta \left(|N_{v,\zeta}||N_s| + |E_{v,\zeta}||E_s|\right)\mathcal{O}_{RL}$. In Alg. 2, the worst case complexity occurs when Alg. 3 is executed for every possible case of (η_m, η_d). That is $\mathcal{O}_{MEDA} = \bar{X}_m \bar{X}_d \mathcal{O}_{NOLA}$, where $\bar{X}_m = \max_\zeta X_{m,\zeta}$ and $\bar{X}_d = \left(\max_\zeta \min\left(X_{d,\zeta}, DB \times |E_{v,\zeta}|\right)\right)$. Consequently, the worst case complexity of Alg. 1 is $\mathcal{O}_{MEDA} \times |h|$.

Algorithm 2: Migration and E2E distance approximation (MEDA)

1 **Input:** A fractional solution of $\tilde{\eta}_x$ ($\tilde{\eta}_m$ or $\tilde{\eta}_d$), optimization problem $\mathcal{P}$, VNF-FG ζ, current cost z
2 **Output:** A Feasible solution of VNF-FG ζ
3 Temporary solution $\mathcal{S} = \emptyset$;
4 **while** *it exists a fractional entry in* $\tilde{\eta}_x$ **do**
5 Find the least fractional entry of $\tilde{\eta}_x \rightarrow \tilde{\eta}_{k_0}$;
6 **if** $\tilde{\eta}_x$ *is* $\tilde{\eta}_d$ ***and*** $k_0 >$ DB $\times\, |E_{v,\zeta}| + 1$ **then**
7 $k_0 =$ DB $\times\, |E_{v,\zeta}| + 1$;
8 Add constraint $\tilde{\eta}_{k_0} = 0$ to $\mathcal{P}$;
9 **if** $\mathcal{P}$ *has a feasible solution* **then**
10 **if** $\tilde{\eta}_x$ *is* $\tilde{\eta}_m$ **then**
11 **if** MEDA$(\tilde{\eta}_d, \mathcal{P}, \zeta, z)$ == **SUCCESS then**
12 Returned solution $\rightarrow \mathcal{S}$ and $z = cost\,(\mathcal{S})$;
13 **continue**;
14 **if** $\tilde{\eta}_x$ *is* $\tilde{\eta}_d$ **then**
15 **if** NOLA$(\mathcal{P}, \zeta, z)$ == **SUCCESS then**
16 Returned solution $\rightarrow \mathcal{S}$ and $z = cost\,(\mathcal{S})$;
17 **continue**;
18 **if** $\tilde{\eta}_x$ *is* $\tilde{\eta}_d$ ***and*** $k_0 >$ DB $\times\, |E_{v,\zeta}|$ **then break** ;
19 **else** Substitute $\tilde{\eta}_{k_0} = 0$ by $\tilde{\eta}_{k_0} = 1$ and solve relaxed $\mathcal{P} \rightarrow \tilde{\eta}_x$;
20 **if** $\mathcal{S} \neq \emptyset$ **then return** (**SUCCESS**, $\mathcal{S}$); ;
21 **else return FAIL**; ;

The A^{2}VF algorithm presented in this section outputs an approximate solution to Problem 1. It requires a central entity collecting available resources from different domains so as to solve the problem as a whole. In practice, domains could be controlled by different operators which are not willing to expose their resources status. Consequently, we propose to

Algorithm 3: Node and Link Assignment (NOLA)

Input: VNF-FG ζ and problem $\mathcal{P}$, current total cost z
Output: A feasible set $\mathcal{S}$ of Φ_n and Φ_e
Temporary solution $\mathcal{S} = \emptyset$;
List of assigned node $v^* = \emptyset$;
foreach *VNF* n'_0 *in VNF-FG* ζ **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 $\Phi_{n_0}^{n'_0} = 1 \rightarrow \mathcal{P}$ and 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 $v^* \leftarrow (n'_0, n_0)$;
foreach *virtual link* e'_0 *in* ζ **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 $\Phi_{e_0}^{e'_0} = 1 \rightarrow \mathcal{P}$ and 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 an acyclic path **p** for e'_0 from $\Phi_e^{e'_0}$;
  foreach *physical link* e_0 *in* ***p*** **do** add $\Phi_{e_0}^{e'_0} = 1$;
  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*** $\text{COST}(\mathcal{S}) < z$ **then return** (**SUCCESS**, $\mathcal{S}$) ;
else return FAIL;

decompose and distribute Problem 1 over the set of operators such that they keep control over their resources and benefit from extending their services through collaboration and parallelizing the optimization process.

V. Decentralized Approximation Algorithm

In this section, a decentralized approximation algorithm is proposed to enable cooperative resource allocation in multi-domain scenarios.

In this work, we propose to illustrate our decentralized approach using an existing multi-domain 5G architecture, currently developed in the 5G-PPP 5GEx project [25]. However, note that this architecture is only an example architecture in which the proposed decentralized resource allocation algorithm can fit: the proposed decentralized approach can apply to other distributed architectures. In this architecture, each operator has its own substrate network which is controlled by one

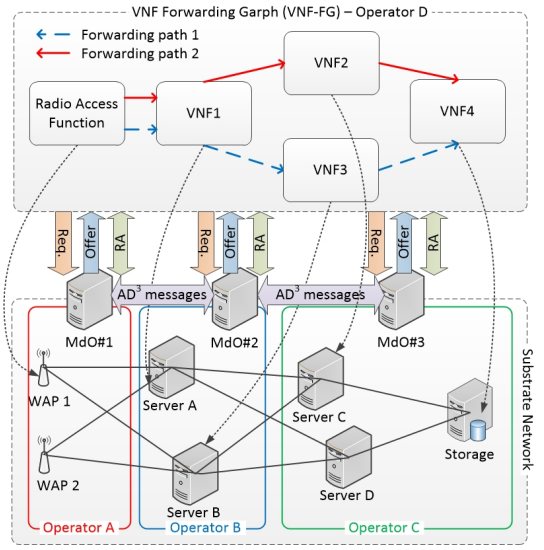


Fig. 2: Decentralized Approximation Algorithm Framework

of several network controllers. Each operator deploys a multi-domain orchestrator to communicate with other operators via inter-operator orchestration API.

A client (e.g. a virtual mobile network operator) requests a VNF-FG from an operator, e.g. Operator B. If the resource of operator B cannot meet the requirements of the client, its multi-domain orchestrator will seek them in its neighbor operators through inter-operator orchestration API. When there is a consensus between operators, operator B can deploy the VNF-FG to serve the client.

A. AD³-based Optimization

5GEx can be an example framework where AD³ can do decentralized optimization. Figure 2 demonstrates operational details of AD³ over multi-domain orchestration framework. The client is a virtual mobile network operator requesting a service which is described as a VNF-FG and each operator (A, B, or C) is able to partially serve its request. First, it sends the request defining the VNF-FG and resources of each VNF to all operators. Then, operators cooperate by exchanging AD³ messages through inter-operator orchestration API. After finishing AD³ process, each operator will send its offers to the client in which determine substrate nodes and links are able to serve VNFs. A decoding algorithm running at the client will take these offers into account to find out a feasible solution. While running the decoding algorithm, resource allocation requests and confirmations are exchanged between the client and operators.

Problem 1 is a NP-hard problem. Although it can be solved by state-of-the-art integer linear programming solvers, it is unfeasible to use them to solve large-scale problems because of the limitations of computation resources and calculation time. As we mentioned in Section II, AD³ is able to provide faster convergence than ADMM and has a library of computationally-efficient factors to tackle hard constraints in a given optimization problem. Thus, one preliminary step before running AD³ is to encode our problem using only such

efficient factors. Later, once AD^3 is executed and provides an output, the solution may not be a feasible solution because AD^3 solves a relaxation of the optimization problem. However, it can be exploited to derive a feasible solution close to the optimal one by a decoding algorithm like the one we present in Section V-C.

Running AD^3, nevertheless, requires synchronization between operators. The benefits of cooperation between mediators (operators in this paper) have been discussed in [39]. AD^3 will shift to next iteration when all nodes finish solving their local problems and exchanging solutions. Let's note that factors and variables are under the responsibility of each operator, without requiring them to disclose their infrastructure properties.

The AD^3-based optimization framework has been discussed. In the following sections, we describe how to formulate the factor graph and propose a heuristic decoding algorithm that is able to convert the output of AD^3 to a feasible solution.

B. Encoding the Optimization Problem to Operate AD^3

In order to adopt AD^3, we have to encode our optimization Problem 1 into a factor graph. Moreover, recall from Section II that our aim is to solely employ computationally-efficient factors, so that the computation of AD^3 as well as the use of messages is efficient. Next, we introduce the factors (functions) from [29] that will allow the encoding of the constraints of Problem 1 as defined in Section III.

Definition 1 (**AtMost1** factor)**.** *It constrains at most one of the variables $x_1, \ldots, x_K$ to be active. Its potential function is defined as:*

$$\theta_{\mathbf{AtMost1}}(x_1, ..., x_K) := \begin{cases} 0 & \text{if } \exists!\ k \text{ s.t. } x_k = 1 \\ & \vee x_1 = ... = x_K = 0 \\ -\infty & \text{otherwise} \end{cases} \tag{12}$$

Definition 2 (**XOR-out** factor)**.** *It constrains at most one of the variables $x_1, \ldots, x_K$ to be active; if one is active, it constrains $x_{K+1} = 1$; if all are inactive, then it constrains $x_{K+1} = 0$. Its potential function is defined as:*

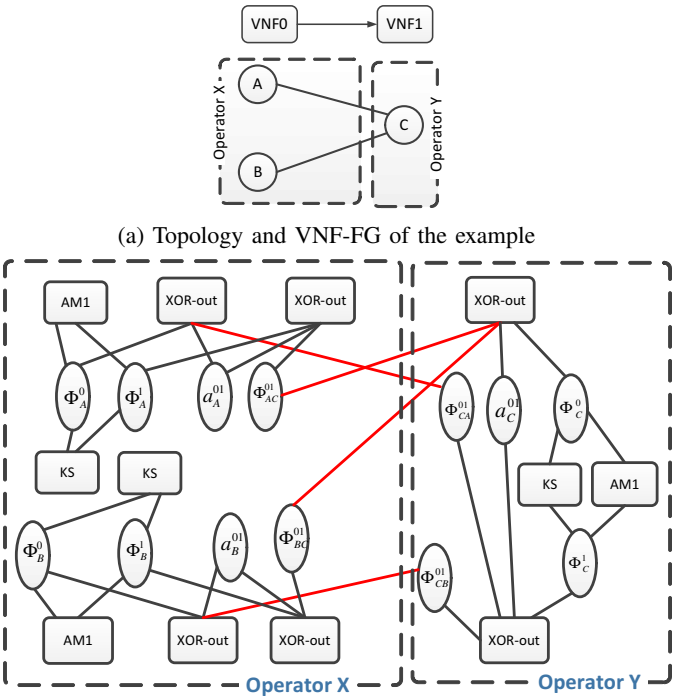
$$\theta_{\mathbf{XOR\text{-}out}}(x_1, \ldots, x_K, x_{K+1}) := \begin{cases} 0 & \text{if } x_{K+1} = 1 \wedge \nexists k \in \{1, \ldots, K\} : x_k = 1 \\ 0 & \text{if } x_{K+1} = 0 \wedge \forall k \in \{1, \ldots, K\} : x_k = 0 \\ -\infty & \text{otherwise} \end{cases} \tag{13}$$

Definition 3 (Knapsack (**KS**) factor)**.** *Its potential function can be defined as:*

$$\theta_{\mathbf{KS}}(x_1, ..., x_K) := \begin{cases} 0 & \text{if } \sum_k x_k \leq C \\ -\infty & \text{otherwise} \end{cases} \tag{14}$$

where C is a given constant.

Now we can encode Problem 1 into a factor graph, as will also be illustrated in Figure 3 using a simple example. Variables, represented as round circles, are linked to the factors representing the hard constraints in the problem, which in turn are represented as rectangles. Each variable contains the value obtained when the variable is active. That value will contribute to the objective value of the optimization problem.



(a) Topology and VNF-FG of the example

(b) AD^3 factor graph of the example

Fig. 3: Example encoding optimization problem

VNF allocation problem can be encoded as follows: constraints (2), (3), (7), (10), and (11) can be encoded by using the **KS** factor; constraints (4) and (1) can be encoded by **AtMost1** factor. Constraint (5) can be rewritten as $\sum_m \Phi_{e^{nm}}^{e^{n'm'}} + \Phi_n^{m'} = \sum_m \Phi_{e^{mn}}^{e^{n'm'}} + \Phi_n^{n'}$. Since each virtual link will be mapped to a single physical path between substrate nodes hosting virtual nodes, we have $\sum_m \Phi_{e^{nm}}^{e^{n'm'}} \leq 1$ and $\sum_m \Phi_{e^{mn}}^{e^{n'm'}} \leq 1$. We introduce an auxiliary binary variable $A_n^{n'm}$ such that $\sum_m \Phi_{e^{nm}}^{e^{n'm'}} + \Phi_n^{m'} = A_n^{n'm'}$ and $\sum_m \Phi_{e^{mn}}^{e^{n'm'}} + \Phi_n^{n'} = A_n^{n'm'}$. These constraints can be encoded by **XOR-out** factors.

Following [29], the complexity of the **AtMost1** and **XOR-out** factors is $O(K \cdot logK)$, where K stands for the number of variables connected to the **XOR** factor. Moreover, according to [40], the complexity of the **KS** factor is linear with the size of the factor. Therefore, we have managed to provide an encoding of our optimization problem that only employs computationally-efficient factors. The convergence speed and complexity of AD^3 has been well studied in [29].

To demonstrate the factor graph and operation of AD^3, we introduce a simple example as shown in Figure 3. A VNF-FG comprising two VNFs is mapped to a substrate network of three nodes A, B, and C. A and B belong to operator X. C is a server of operator Y. Constraints (2), (5), and (4) are described as factors **KS**, **XOR-out**, and **AtMost1** (noted AM1) respectively in Figure 3b. Red links presents

the connections between factors and variables belonged to different operators, called as inter-connections. At the end of each iteration, the solutions of variables of inter-connections will be exchanged between two operators and utilized as the input of the next iteration. To form the factor graph, each operator should be aware of its connections to other operators, however it is not necessary to know current status of other operators (available bandwidth, free storage, etc.). That addresses a practical scenario where operators want to cooperate to provide a service to a given client, but they do not want to expose their current resources to other competitors.

C. Heuristic Decoding Algorithm

Once executed, AD^3 finally requires decoding of the solution. We propose Algorithm 4, which is a lightweight decoding algorithm to decode AD^3 solution to feasible VNF-FG allocation. This algorithm will be executed by the VNF-FG requester after collecting all solutions from other operators. The algorithm begins with rounding the most fractional entries among $\Phi_n^{n'}$ to 1. The VNF-FG owner checks feasibility by sending resource allocation requests (as shown in Fig. 2) to the owner of the substrate node (Operator A, B, or C in Fig. 2). The owner of substrate node answers with an acknowledgment when the substrate node has sufficient resources to host the VNF. Otherwise, it sends a negative-acknowledgment and the VNF-FG owner fixes $\Phi_n^{n'}$ to 0. Note that we assume substrate node owners are "cooperative" in this paper. It means they always allow to allocate VNFs whenever the substrate node has sufficient resources. Non-cooperative behaviors of operators will be considered in future studies. The algorithm repeats the rounding process until all VNFs are assigned. Then, finding a physical path for each virtual link could be done by adopting a process presented in Algorithm 3. A successful mapping between virtual links and physical paths grants resources for the VNF-FG. The complexity of Alg. 4 is similar to the complexity of NOLA which is polynomial time.

Algorithm 4: Decoding algorithm

Input: VNF-FG ζ, AD^3 solution $\tilde{\Phi}_n^{n'}$
Output: A feasible set $\mathcal{S}$ of Φ_n and Φ_e
foreach *VNF* $n_0^{'}$ *in VNF-FG* ζ **do**
  if $\exists\tilde{\Phi}_n^{n_0^{'}} == 1$ **then** $\Phi_n^{*n_0^{'}} \leftarrow \Phi_n^{n_0^{'}}$ and **continue** ;
  else
    while $\exists\tilde{\Phi}_n^{n_0^{'}} \notin \{0,1\}$ **do**
      Find the most fractional entry $\Phi_{n_0^{'}}^{n_0^{'}}$;
      Check feasibility of assigning $n_0^{'}$ to n_0;
      if *assignment is feasible* **then** $\Phi_n^{*n_0^{'}} \leftarrow 1$;
      else $\Phi_n^{*n_0^{'}} \leftarrow 0$;
    if $\nexists\Phi_n^{*n_0^{'}} == 1$ **then return FAIL**;
foreach *virtual link* $e_0^{'}$ *of* ζ **do**
  Set of candidate links $\mathcal{L}_{e_0^{'}} = \emptyset$;
  foreach $\tilde{\Phi}_e^{e_0^{'}} > 0$ **do** $\mathcal{L}_{e_0^{'}} \leftarrow e$;
  while *true* **do**
    Find an acyclic feasible path $\mathbf{p}$ connects all $e_0^{'} \in \mathcal{L}_{e_0^{'}}$;
    if $\nexists\mathbf{p}$ **then return FAIL**;
    else break ;

VI. Numerical Results

A. Centralized Optimization

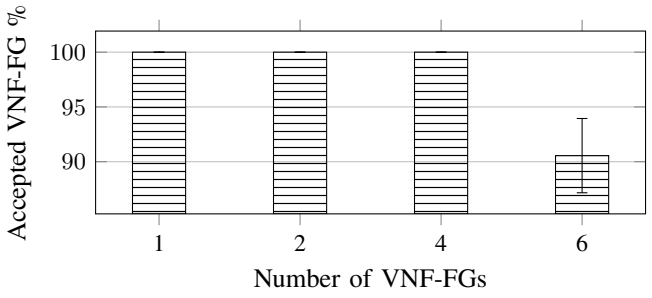
In this section, the performance of A^{2}VF and ILP-based approaches are compared to highlight the pros and cons of A^{2}VF . Two objective functions are formulated as an ILP problem: with migration and end-to-end distance costs and without them - ILP-ND (no distance). ILP-ND is the upper-bound of the performance presented in [7]. ILPs are solved by Gurobi® which is deemed as one of the most advanced ILP solvers nowadays. Note that ILP solvers are considered as reference for performance comparison of our proposed heuristic algorithm.

In this paper, we consider a number of VNF-FGs in the time window t. The placement of VNFs in the previous window $t-1$ is assumed to be known. The number of VNFs in each VNF-FG and the resources of substrate nodes are generated arbitrarily for simulation. 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, which is initialized to 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. Each scenario is run 30 times with different seeds of the number of VNFs in VNF-FG and their resource requirements, and then, the results are the average of them. A 95% confidence interval is used in this paper.

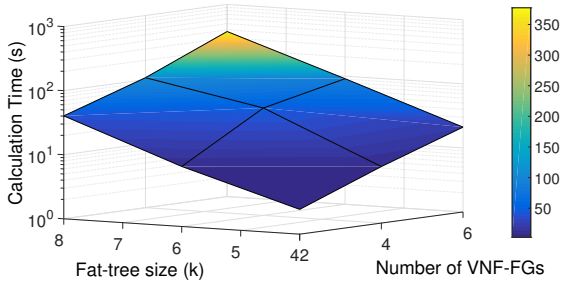
We opted for the same simulation parameters as in [7]. The number of VNFs in each request is selected randomly in the range $[3, 6]$. The computational, storage, and memory requirements of each VNF-FG request are uniformly distributed in the range $[25, 30]$ while the link requirements are in the range $[55, 60]$. 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 in order to optimize the migration distance. Referring to [37], 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 favor a low migration distance.

1) Performance: We first look at the number of accepted VNF-FG requests successfully allocated, calculation time, migration distance and transmission distance with different fat-tree size and number of VNF-FGs. The ILP-based approach provides the optimal solution and is considered as a reference. Figure 4a shows the percentage of accepted VNF-FGs obtained by A^{2}VF when the number of VNF-FGs is 1 to 6 and the fat-tree size is 4. The acceptance rate of ILP-based approach is 100% for all cases. The acceptance rate of A^{2}VF is 100% when the number of VNF-FG is 1. It drops to 90% when the number of VNF-FGs is 6, i.e., 10% less than the optimal solution.

The calculation time of A^{2}VF with different number of

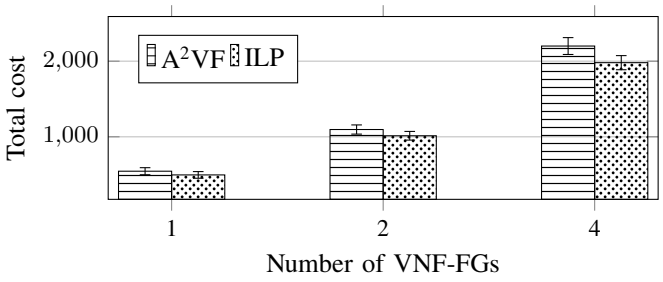


(a) A^{2}VF acceptance rate

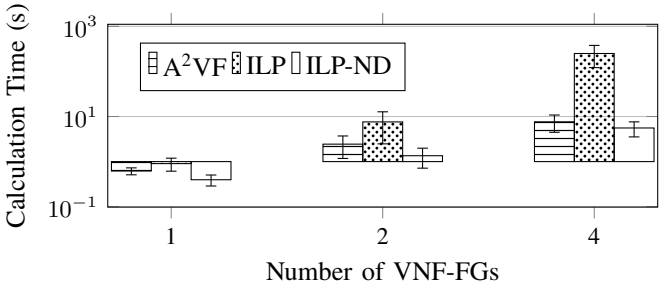


(b) A^{2}VF calculation time

Fig. 4: Performance of A^{2}VF with infinity DB



(a) Average total cost of A^{2}VF and ILP

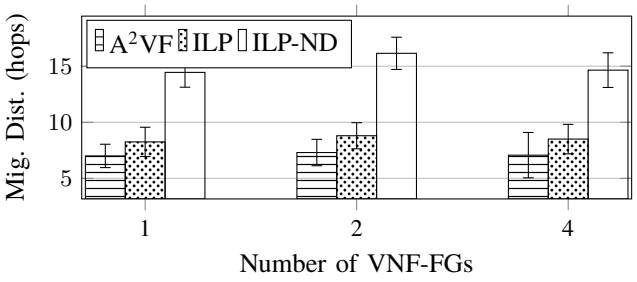


(b) Calculation time of A^{2}VF and ILP

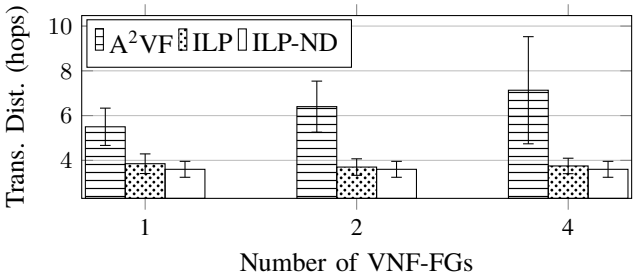
Fig. 5: Performance of A^{2}VF and ILP with infinity DB

VNF-FGs and fat-tree sizes is shown in Figure 4b. Generally, the calculation time decreases when the number of VNF-FGs or the fat-tree size decreases. The maximum calculation time, 400 seconds, occurs when there are 8 VNF-FGs in 8-ary fat-tree size substrate network.

In the following simulations, the fat-tree size is fixed at 4. Figure 5a 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 VNF-FGs is 1 or 2



(a) Migration distance



(b) Transmission distance

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

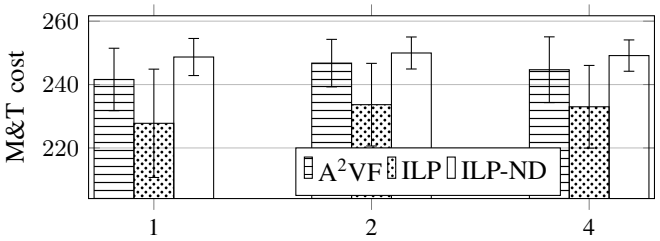


Fig. 7: Migration cost and transmission cost of A^{2}VF , ILP, and ILP-ND

and becomes greater when the number of VNF-FGs increases. At 4 VNF-FGs, the total cost of A^{2}VF is about 10% greater than that of ILP.

Figure 5b shows that A^{2}VF outperforms ILP-based approaches in terms of calculation time. ILP-ND does not consider migration 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 a feasible solution for 4-ary fat-tree and 1 to 4 VNF-FGs. Figure 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 cost (M&T) is the lowest as shown in Figure 7. M&T cost of A^{2}VF is higher than ILP, but it is still less than that of ILP-ND.

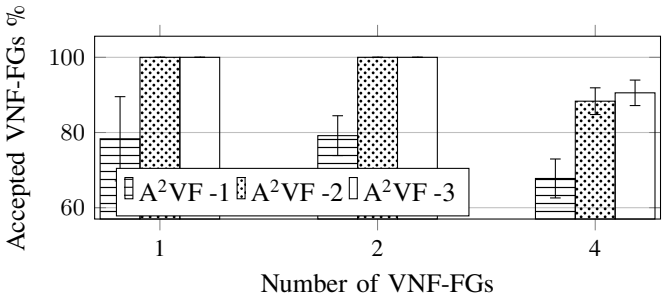
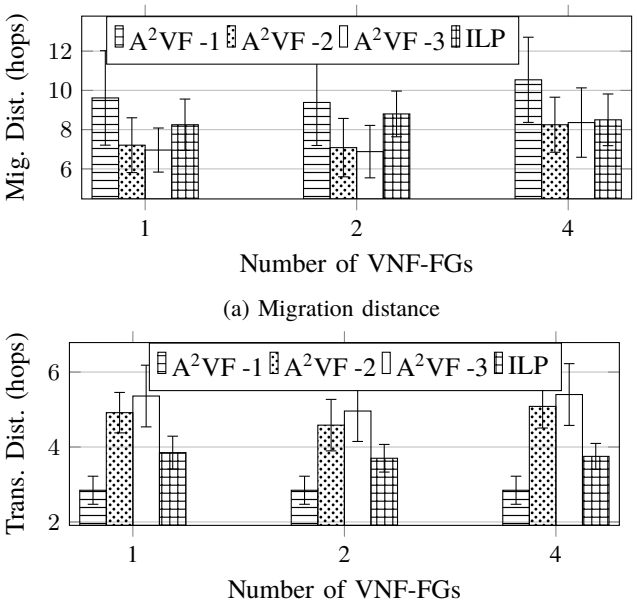


Fig. 8: Average VNF-FG acceptance rate of ILP and A^{2}VF under different distance bounds

2) 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 VNF-FGs is from 1 to 4. DB varies from 1 to 3.

The percentage of accepted VNF-FGs is given in Figure 8. When DB is 1, each virtual link only uses a substrate link. Due to this strictness, the number of accepted VNF-FGs is low. Increasing DB could help to increase the acceptance ratio, especially when the number of VNF-FGs is large. When the number of VNF-FGs 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 Figure 4a which is 100% when the number of VNF-FGs is 4.

Figure 9 shows the average migration and transmission distance of ILP and A^{2}VF with DB from 1 to 3 denoted as A^{2}VF -1,A^{2}VF -2, and A^{2}VF -3 respectively. When DB increases, the range of η_d values are extended. Consequently, the transmission distance could be extended in order to have the better migration distance. The total costs are shown in Figure 10. The total cost with DB $= 2$ and DB $= 3$ are similar. However, there are remarkable gaps between A^{2}VF -1 and A^{2}VF -2 and A^{2}VF -3. This is because the very short DB prevents the algorithm from reaching the better solutions. The gap between optimal solution and A^{2}VF could be shortened by setting DB to infinity as shown in Figure 5a.



(a) Migration distance

(b) Transmission distance

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

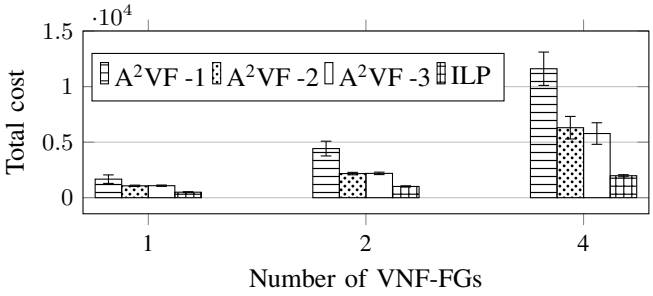


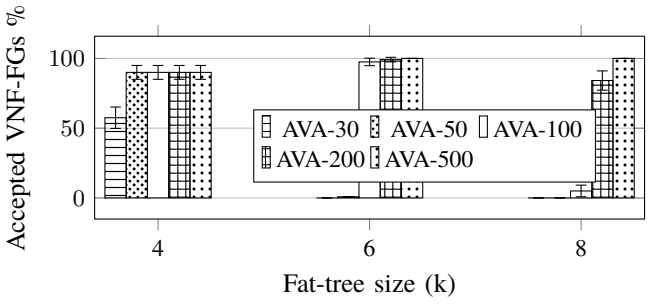
Fig. 10: Average total cost of ILP and A^{2}VF under different distance bounds

B. Decentralized Optimization

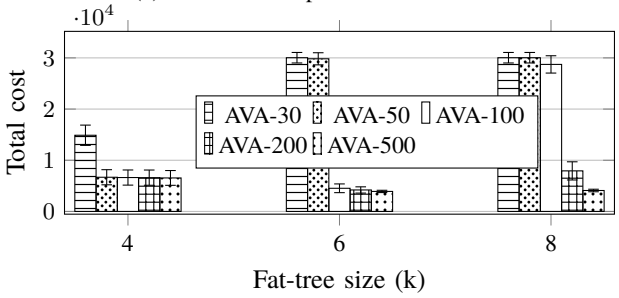
The performance of the proposed decentralized optimization approach AD^3-VA and the centralized optimization approach A^{2}VF are compared in this section to determine the pros and cons of AD^3-VA. The AD^3 library used in this paper was downloaded from the repository of AD^3 [41]. The current version does not support parallel computation. Consequently, even though the problem was broken into sub-problems as described in Section V-B, the sub-problems are solved sequentially leading to the high computation time. It is obvious to expect a significantly better calculation time when sub-problems are solved simultaneously. Furthermore, transmission latency of exchanging AD^3 messages between multi-domain orchestrator is assumed to be negligible thanks to the low latency of optical backbone network [42]. For instance, the latency between data centers 20 km apart is 110 μs, thus it can cost up to $55ms$ for 500 iterations.

AD^3-VA is configured with various numbers of iterations $30, 50, 100, 200, 500$ and denoted as AVA-xx, where xx is the number of iterations. We first consider the rate of successfully allocated VNF-FG requests when the fat-tree size varies ($k = 4, 6, 8$) and the number of VNF-FGs is 6. The acceptance rate of AD^3-VA under different fat-tree sizes is presented in Figure 11a. The larger fat-tree size has more nodes, therefore the complexity of resource allocation increases. Consequently, it requires a greater number of iterations to determine a solution. The small number of iterations provides a low quality input for the decoding algorithm, therefore leading to a low acceptance rate.

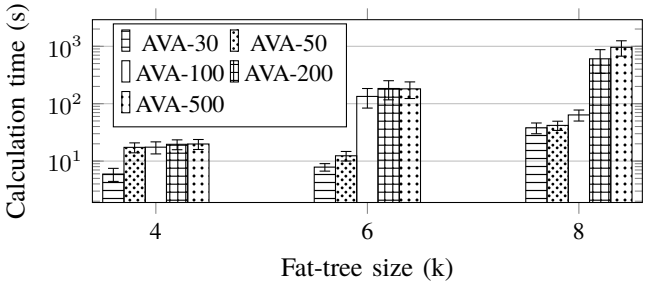
The average total cost under various fat-tree sizes is shown in Figure 11b. With small number of iterations, AD^3-VA is unable to provide a good solution due to the low quality of AD^3 output. When the number of iterations increases, the



(a) VNF-FG acceptance rate of AD^3-VA

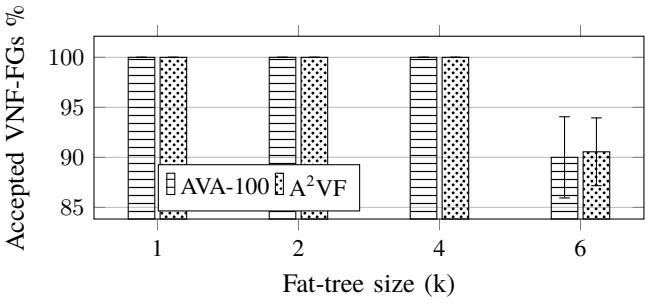


(b) Total cost of AD^3-VA

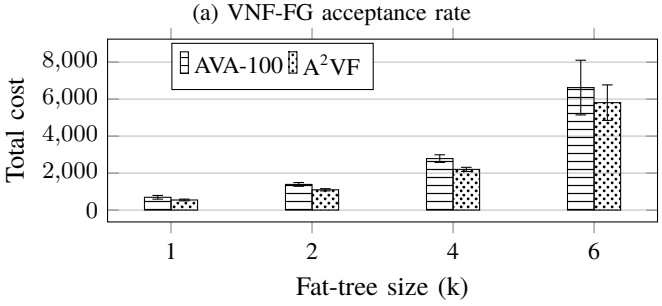


(c) Calculation Time of AD^3-VA

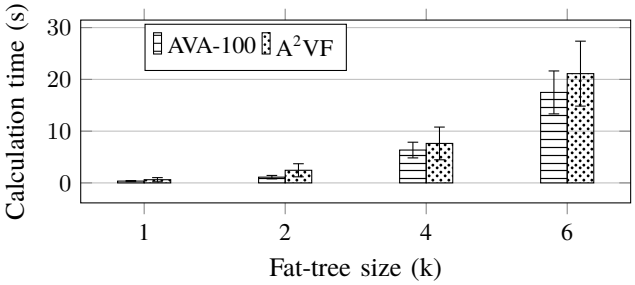
Fig. 11: Performance of AD^3-VA– 6 VNF-FGs



(a) VNF-FG acceptance rate



(b) Total cost



(c) Calculation Time

Fig. 12: Performance of AD^3-VA-100 vs A^{2}VF

quality of output of AD^3 is enhanced, thus reducing the cost.

Figure 11c shows the calculation time of AD^3-VA under different fat-tree sizes $(4, 6, 8)$. Generally, when the fat-tree size increases, the calculation time increases due to the increase of complexity. When the fat-tree size is 4 or 6, the calculation time of AVA-200 and AVA-500 are similar. Although the time for AD^3 process of AVA-200 is shorter than AVA-500, the time for decoding the solution of AVA-200 is longer than AVA-500 because of its lower AD^3 output. However, when the fat-tree size is 8, even AVA-500 does not have a good AD^3 output. That leads to the decoding time of AVA-500 as long and the gap in calculation time between AVA-200 and AVA-500 as significant.

The next simulation is to compare the performance of AD^3-VA and A^{2}VF under varying number of VNF-FGs. The number of iterations is 100 and the fat-tree size is 4. First, the number of accepted VNF-FG requests is considered. Figure 12a shows that there is no difference in the acceptance rate when the number of VNF-FGs varies from 1 to 4 and A^{2}VF is slightly better than AVA-100 when the number of VNF-FGs is 6. However, the total cost in Figure 12b confirms the better solution obtained by A^{2}VF . The gap between AD^3-VA and A^{2}VF is up to 20%. The calculation time of AVA-100, nevertheless, is shorter than A^{2}VF as shown in Figure 12c.

VII. 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 dynamics and reallocation ability has not been addressed. In this paper, the impact of reallocation of VNF-FGs was studied and a model considering dynamic behavior was provided. To justify the cost of reallocation, the migration distance and transmission distance were considered. The optimization 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 adaptation of DB parameter could be considered in future work.

We also provided a decentralized allocation algorithm (namely AD^3-VA) that enables operators to allocate virtual network functions in large-scale scenarios in a decentralized and cooperative manner, whilst keeping information about infrastructure under control of the operators themselves. In return, AD^3-VA has worse performance than A^{2}VF , however the gap is less than 20% in total cost and there is no difference in VNF-FG acceptance ratios (in 4-ary fat tree). These results motivate possibilities of adopting AD^3 and advancements in distributed optimization in networking. In future, we will study the extension of our work to consider fog computing and mobile edge computing. There distributed solution can also be helpful, but the system model needs to be adapted to consider limited resources of edge nodes and their connectivity. Also, a distributed scheme that tackles selfish operators (non-cooperative distributed optimization) can unveil new applications of this work.

References

[1] A. Nakao and P. Du, "Application-specific slicing for mvno and traffic characterization," *J. Opt. Commun. Netw.*, vol. 9, no. 2, pp. A256–A262, 2017.

[2] C. Liang and F. R. Yu, "Wireless virtualization for next generation mobile cellular networks," *IEEE Wireless Commun.*, vol. 22, no. 1, pp. 61–69, 2015.

[3] A. Nakao, P. Du, and T. Iwai, "Application specific slicing for mvno through software-defined data plane enhancing sdn," *IEICE Trans. Commun.*, vol. 98, no. 11, pp. 2111–2120, 2015.

[4] 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.

[5] 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.

[6] 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.

[7] R. Riggio, A. Bradai, D. Harutyunyan, T. Rasheed, and T. Ahmed, "Scheduling wireless virtual networks functions," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 2, pp. 240–252, June 2016.

[8] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. of CloudNet*, Oct 2014, pp. 7–13.

[9] S. Sahhaf, W. Tavernier, M. Rost, S. Schmid, D. Colle, M. Pickavet, and P. Demeester, "Network service chaining with optimized network function embedding supporting service decompositions," *Comput. Netw.*, vol. 93, pp. 492–505, 2015.

[10] H. Moens and F. D. Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *Proc. of IEEE CNSM*, Nov 2014, pp. 418–423.

[11] Z. Ye, X. Cao, J. Wang, H. Yu, and C. Qiao, "Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization," *IEEE Netw.*, vol. 30, no. 3, pp. 81–87, May 2016.

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

[13] M. M. Tajiki, S. Salsano, M. Shojafar, L. Chiaraviglio, and B. Akbari, "Joint energy efficient and qos-aware path allocation and vnf placement for service function chaining," *CoRR*, vol. abs/1710.02611, 2017.

[14] ——, "Energy-efficient path allocation heuristic for service function chaining," in *Proc. of ICIN, Paris, France*, 2018, pp. 20–22.

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

[16] M. Mechtri, C. Ghribi, and D. Zeghlache, "A scalable algorithm for the placement of service function chains," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 533–546, 2016.

[17] J. G. Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 518–532, 2016.

[18] R. Mijumbi, J. Serrat, J. l. Gorricho, S. Latre, M. Charalambides, and D. Lopez, "Management and orchestration challenges in network functions virtualization," *IEEE Commun. Mag.*, vol. 54, no. 1, pp. 98–105, January 2016.

[19] R. Munoz, R. Vilalta, R. Casellas, R. Martinez, T. Szyrkowiec, A. Autenrieth, V. Lopez, and D. Lopez, "Integrated SDN/NFV management and orchestration architecture for dynamic deployment of virtual SDN control instances for virtual tenant networks," *IEEE J. Opt. Commun. Netw.*, vol. 7, no. 11, pp. B62–B70, November 2015.

[20] J. R. Soares, C. Gonçalves, B. Parreira, P. Tavares, J. Carapinha, J. Barraca, R. Aguiar, and S. Sargento, "Toward a telco cloud environment for service functions," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 98–106, February 2015.

[21] S. Hong, J. P. Jue, Q. Zhang, X. Wang, C. She, H. C. Cankaya, and M. Sekiya, "Effective virtual optical network embedding based on topology aggregation in multi-domain optical networks," in *Proc. of IEEE OFC*, March 2014, pp. 1–3.

[22] Q. Zhang, X. Wang, I. Kim, P. Palacharla, and T. Ikeuchi, "Service function chaining in multi-domain networks," in *Proc. of IEEE OFC*, 2016, pp. 1–3.

[23] A. Francescon, G. Baggio, R. Fedrizzi, R. Ferrusy, I. G. B. Yahiaz, and R. Riggio, "X–MANO: Cross–domain management and orchestration of network services," in *IEEE NetSoft*. IEEE, 2017, pp. 1–5.

[24] G. Baggio, A. Francescon, and R. Fedrizzi, "Multi-domain service orchestration with X-MANO," in *IEEE NetSoft*, July 2017, pp. 1–2.

[25] 5GEx project. [Online]. Available: http://www.5gex.eu/

[26] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[27] Z. Allybokus, K. Avrachenkov, J. Leguay, and L. Maggi, "Lower bounds for the fair resource allocation problem," *SIGMETRICS Perform. Eval. Rev.*, vol. 45, no. 3, pp. 167–173, Mar. 2018.

[28] Q. Pham Tran Anh, "Algorithms and optimization for quality of experience aware routing in wireless networks : from centralized to decentralized solutions," Theses, Université Rennes 1, Jan. 2017.

[29] A. Martins, F. Mário, A. Pedro, N. A. Smith, and E. P. Xing, "AD^3: Alternating directions dual decomposition for map inference in graphical models," *J. Mach. Learn. Res.*, vol. 16, pp. 495–545, 2015.

[30] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, "Tree-reweighted belief propagation algorithms and approximate ml estimation by pseudo-moment matching," in *Proc. of AISTATS*, vol. 21, 2003, p. 97.

[31] N. Komodakis, N. Paragios, and G. Tziritas, "MRF optimization via dual decomposition: Message-passing revisited," in *Proc. of IEEE ICCV*, 2007, pp. 1–8.

[32] A. Globerson and T. S. Jaakkola, "Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations," in *Proc. of NIPS*, 2008, pp. 553–560.

[33] T. Hazan and A. Shashua, "Norm-Product Belief Propagation: Primal-Dual Message-Passing for Approximate Inference," *IEEE Trans. Inf. Theory*, vol. 56, no. 12, pp. 6294–6316, Dec 2010.

[34] "Network Functions Virtualisation: Use cases," *ETSI GS NFV*, vol. 1, p. V1, 2013.

[35] M. Mechtri, C. Ghribi, O. Soualah, and D. Zeghlache, "NFV Orchestration Framework Addressing SFC Challenges," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 16–23, 2017.

[36] "Network Functions Virtualisation (NFV): Architectural framework," *ETSI GS NFV*, vol. 2, no. 2, p. V1, 2013.

[37] 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.

[38] N. Megiddo, *On the complexity of linear programming*. IBM Thomas J. Watson Research Division, 1986.

[39] T. Penya-Alba, M. Vinyals, J. Cerquides, and J. A. Rodriguez-Aguilar, "Exploiting Max-sum for the Decentralized Assembly of High-valued Supply Chains," in *Proc. of AAMAS*, 2014, pp. 373–380.

[40] M. B. Almeida and A. F. Martins, "Fast and robust compressive summarization with dual decomposition and multi-task learning." in *ACL (1)*, 2013, pp. 196–206.

[41] *AD^3* github. [Online]. Available: https://github.com/andre-martins/AD3

[42] "Low Latency – How low can you go?" Infinera, Tech. Rep., 2016.