




Network Function Virtualization Resource Allocation Based on Joint Benders Decomposition and ADMM

Ye Yu , *Student Member, IEEE*, Xiangyuan Bu , Kai Yang , *Member, IEEE*, Hung Khanh Nguyen, and Zhu Han, *Fellow, IEEE*

Abstract—Network function virtualization (NFV) has emerged as a new technology to reduce the cost of hardware deployment. It is an architecture that using virtualized functions run on the virtual machine to achieve services instead of using specific hardware. Although NFV brings more opportunities to enhance the flexibility and efficiency of the network, resource allocation problems should be well taken into consideration. In this paper, we investigate the virtual network function (VNF) resource allocation problem to minimize the network operation cost for different services. Both setting the VNF instances for each virtual machine and allocating the traffic volume in the network are considered. The problem is formulated as a mixed integer programming problem. Although it can be solved in a centralized fashion which requires a central controller to collect information from all virtual machines, it is not practical for large-scale networks. Thus, we propose a distributed iteration algorithm to achieve the optimal solution. The proposed algorithm framework is developed based on the joint Benders decomposition and alternating direction method of multipliers (ADMM), which allows us to deal with integer variables and decompose the original problem into multiple subproblems for each virtual machine. Furthermore, we describe the detail implementation of our algorithm to run on a computer cluster using the Hadoop MapReduce software framework. Finally, the simulation results indicate the effectiveness of the algorithm.

Index Terms—Network function virtualization, resource allocation, Benders decomposition, alternating direction method of multipliers, Hadoop, MapReduce.

I. INTRODUCTION

NETWORK function virtualization (NFV) is a currently popular topic among the research issues of industry and academia. In the past, the function of the network is achieved by the middleboxes or hardware appliances, such as firewall, proxies, load balancers, and so on [1]. NFV provides an opportunity

to reduce costs of implementing expensive hardware and flexibility for the network service, which means the instances of virtualization network functions (VNFs) can be created, migrated and released in any circumstances. The most significant power consumptions in the NFV are capital expenditures (CAPEX) and operating expenditures (OPEX) [2]. NFV is achieved on Virtual Machines (VMs) that are located in databases. Computation, storage, and network resources are required for the implementation of VNFs. The operators can provide specific service by composing service function chain (SFC) using VNFs. However, the SFC should meet the users' requirement considering both the QoS parameters and the cost.

NFV operation is supported by network functions virtualization infrastructure (NFVI), which is composed of hardware and software to enable the physical and virtual layer of the network. Each VNF needs to be associated to a specific VM. Creation and operation of VNFs are dominated by the management and orchestration (MANO). Meanwhile, software-defined networking (SDN) is integrated with NFV in the architecture [3]. The benefit of introducing SDN is separating the control and data plane by programming. This will bring flexibility to the network, and the forwarding rules can be easily implemented by the SDN controller. The SDN-NFV architecture is a trend, and many organizations have proposed standards and related research [4].

NFV has been a promising technology in the vehicular network, especially in efficient traffic management, road safety, and entertainment. In the next generation network, vehicle applications will require ultra-reliable low-latency communications. Many NFV frameworks are proposed for serving vehicular network so far. Enabling NFV on top of vehicles will provide mobile service providers an easier access to the emerging paradigms. Currently, vehicle-to-everything (V2X) communication is a hot topic in the vehicle communication system. The SDN-NFV architecture has many merits, along with V2X. For example, it will be easier to acquire dynamic data traffic routing for vehicles with high mobility. In other words, the waiting time of traffic light control or congestion can be significantly reduced. Because the virtual machines are virtualized and migrated among vehicles, the specific vehicle can remain its operating area by exchanging the information involves computation, storage, sensing, communication resources, and environmental conditions. With timely service based on NFV, the accident rescue will be more efficient with rapid emergency response. In addition, using NFV technique can reduce fuel consumption and carbon emissions with optimized on-road strategy. In short, introducing

Manuscript received December 25, 2018; revised June 2, 2019 and October 15, 2019; accepted December 4, 2019. Date of publication December 12, 2019; date of current version February 12, 2020. This work was supported by the National Natural Science Foundation of China under Grant 61771054 and Grant 61501028 and in part by the US MURI AFOSR MURI 18RT0073, NSF EARS-1839818, CNS1717454, CNS-1731424, CNS-1702850, and CNS-1646607. The review of this article was coordinated by Prof. R. Jantti. (*Corresponding author: Kai Yang.*)

Y. Yu, X. Bu, and K. Yang are with the School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China (e-mail: yuye@bit.edu.cn; bxy@bit.edu.cn; yangkai@ieee.org).

H. K. Nguyen is with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004 USA (e-mail: hknguyen9@uh.edu).

Z. Han is with the University of Houston, Houston, TX 77004 USA, and also with the Department of Computer Science and Engineering, Kyung Hee University, Seoul 446-701, South Korea (e-mail: zhan2@uh.edu).

Digital Object Identifier 10.1109/TVT.2019.2959347

NFV in vehicle communication network can provide advanced services to the end users on the road, and it can improve the QoS of the current traffic techniques.

Although NFV has many promising advantages, there are some problems which cannot be ignored. The resource allocation problem with SFC is an intractable issue in operating virtualization networks. VMs are virtualized on the physical machine, and they need to be appropriately assigned. Furthermore, the resources for the network are limited, which need to give an optimal decision on allocating power, bandwidth, etc. [5], [6]. In the general cases, the global optimal solution is difficult to obtain. This is due to the formulated problems often are mixed-integer nonlinear problems. It is hard to solve such NP-hard problems, not mentioning the influence of the network scale [7]. Thus, the heuristic algorithm is often proposed, e.g., in [8] and [9].

In the service requests, some are related to the distributed resources. NFV has a distributed nature because the virtualization of the network is unrestricted from the position of the physical machines [10]. Many distributed algorithms are introduced to this area, such as game theory, alternating direction method of multipliers (ADMM), and the other decomposition algorithms [1]. The distributed architecture makes full utilization of the resource in the networks.

The resource allocation problem for NFV is studied by many papers. However, the global optimal solution is difficult to obtain in many cases, considering a large number of constraints and the large-scale of the network which will bring computing ability issues to the central controller. Furthermore, the proposed algorithms may have many constraints, which are not generally suitable for certain other cases. These challenges motivate us to propose a general method to acquire the global optimal solution for the NFV resource allocation. For the scale problem of the network, it is highly desirable to solve the problem in a distributed fashion to make full use of the resource allocated to VMs. In this way, the computation task for the controller can be reduced significantly, as well as the execution time for the network.

In this paper, we consider the NFV resource allocation problem, including the VNF placement problem and traffic management problem. We aim to minimize the costs of implementing VNFs and operating traffic in NFV networks. The problem is formulated as a mixed integer linear programming (MILP) problem, and it is solved in a distributed manner by joint Benders decomposition and ADMM. We also describe the implementation details of our algorithm using the Hadoop MapReduce software framework. The simulation results verify the merits and properties of our algorithm.

Our main contributions are summarized as follows:

- We formulate the resource allocation problem in an NFV network as a mixed integer linear problem, which aims to minimize the system cost. Both VNF placement and traffic management problems are considered.
- We first propose an algorithm based on joint Benders decomposition and ADMM. The algorithm can be applied to mixed integer problems and solve it in a distributed manner as a general framework. The Benders decomposition can separate the integer variables apart with continuous

TABLE I
LIST OF NOTATIONS

Symbol	Definition
N	Total number of VMs
M	Total number of SFC
K	Total number of functions in SFC
F	VNF set for the network
l	Loop index number for Benders decomposition
f_k^m	k^{th} Function in m^{th} SFC
$\delta_{n,k}^m$	Binary variable denoting existence of the VNF f_k^m for n^{th} VM
e_k^m	Total number of the instances of f_k^m
$v_{n,k}^m$	Continuous variable denoting traffic volume for f_k^m
$u_{n,k}^m$	Global copy of $v_{n,k}^m$
ξ_k^m	Cost for implementing one instance of f_k^m
γ_k	Cost coefficient for unit traffic of VNF k
ϕ_k	Resource cost for the unit traffic of VNF k
μ_{f_k}	Input-output ratio of VNF k
C_n	Capacity for VM n
Q_p	Cost for VNF placement
Q_t	Cost for traffic
Q_{total}	Total cost for the network
R_k	Request data rate for VNF k
R	Request data rate for SFC
$U_{\delta_{n,k}^m}$	Upper bound of $\delta_{n,k}^m$
$L_{\delta_{n,k}^m}$	Lower bound of $\delta_{n,k}^m$
α	Optimal value of the subproblem
α^{down}	Lower bound of α
$\theta^m(l)$	Dual variable for the constraints that fixing the binary variables' value
Q_{up}^l	Upper bound of Q_{total}
Q_{down}^l	Lower bound of Q_{total}
ε	Pre-defined tolerance
λ	Lagrangian multiplier
ρ	Penalty parameter

variables. For the subproblem, a large-scale of continuous variables can be handled in a distributed way using ADMM. In this paper, we solve the VNF placement problem in the outer loop of the algorithm. For the traffic management problem with continuous variables, we use ADMM to solve it distributedly.

- We evaluate the performance of our proposed algorithm in the simulation section. The implementation of our algorithm in the Hadoop MapReduce is described in detail. The results indicate that our algorithm is efficient for large-scale mixed integer problems and the computation complexity is significantly reduced.

The rest of the paper is organized as follows. In Section II, the related work is surveyed. The NFV system model and the resource allocation problem are presented in the Section III. Then, the algorithm based on joint Benders decomposition and ADMM is the proposed in Section IV. Section V describes the implementation of the proposed algorithm using the Hadoop MapReduce. The simulation results are shown in Section VI-A, and conclusions are stated in Section VII. The list of notations of this paper is given in Table I.

II. RELATED WORK

NFV is a new network architecture, as shown in Fig. 1, proposed by the industry for the software-driven network [9].

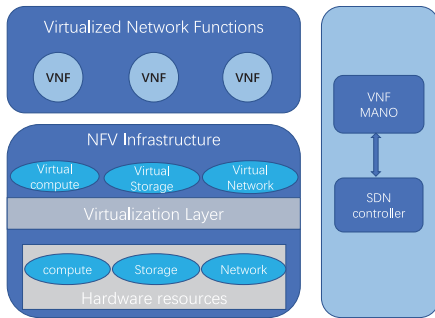


Fig. 1. The NFV network architecture.

The up to date developments and challenges of NFV is surveyed in [11]. In [12], a new model for enhancing virtual machine applications performance is proposed. In [13], the authors propose an architecture involving both mobile edge computing (MEC) and NFV, which enables joint managing applications and virtual network functions. The NFV compound effect of the system is evaluated. An adaptive memory load management scheme for the server running on the virtual machine in the cloud data center is proposed in [14], which can prevent the memory overload. In [15], the authors jointly minimize the power consumption considering servers and switches, which generates the optimal VNFs placement. In [16], the authors modeled the VNF allocation problem by queuing system with constraints and solve the Markov decision process to find the optimal policy. In [17], a new dynamic resource pooling and trading mechanism in network virtualization are proposed and solved by the Stackelberg game. Most of the literature solved the problems using heuristic methods which cannot guarantee the global optimal.

Nowadays, NFV/SDN architecture is popular for its flexibility. In [18], the software-defined and network virtualization networking is surveyed, as well as the SDN applied on OpenADN in a multi-cloud environment. The SDN-NFV mobile networks are modeled in [19] to minimize equipment failure risks as well as fast recovery of network nodes after a disaster happens. A service function chaining embedding problem for SDN-NFV is presented in [20], which is solved by splitting the traffic to several VNFs in the service chain. In [21], the authors introduce SDN and NFV including service chains. SFC is introduced in the NFV, which connects several services and enables a single network to achieve many services. Clustered NFV service chaining is adopted to obtain the optimal number of clusters in [22], which can minimize the end-to-end service time in MEC RANs. The function chains can be customized which is stated in [23]. Vehicles are playing a major role in the distributed mobile scenario. In [24], the authors propose an intelligent VNFs selection strategy in Vehicular Cloud Network (VCN), which utilizes deep neural network (DNN) and Multi-Grained Cascade Forest (gcForest) to distinguish service behaviors. In [25], the authors elaborate the potential use of the NFV service for the heterogeneous vehicles with many benefits. In [26], the author propose a theoretical framework to evaluate the performance of a Long-Term Evolution (LTE) virtualized mobility management entity (vMME) hosted in a data center. In [27], the authors

discuss the new trends of applying NFV in 5G network to manage the wireless/mobile broadband (5G WMB).

Distributed solutions become more and more important since the networks become denser and denser [28]–[30]. There are plenty of algorithms have been taken into consideration in the distributed applications. In [31], distributed algorithms for big data applications are introduced. Game theory is widely used in resource allocation of distributed networks. In [32], the authors analyze the congestion mitigation problem in NFV for both centralized and distributed methods using game theory. A Stackelberg game is adopted as the approach to solve the resource management problem in LTE unlicensed in [33]. In [34], a hierarchical game approach is presented for generating optimal strategies of visible light communication and D2D heterogeneous network in a distributed manner. Benders decomposition is first proposed in [35] and it is well studied by many researchers. In [35], Benders decomposition is introduced in details. The market risk management problem is modeled as a stochastic linear complementarity problem in [36], and Benders decomposition is adapted to this equilibrium models. Benders decomposition is suitable for solving mixed integer programming. In [37], a joint base station (BS) association and power control algorithm is proposed based on Benders decomposition. The size and schedule in microgrids operation are studied using Benders decomposition in [38]. However, most of the papers only provide centralized solutions for the mixed integer programming. In addition to game theory and Benders decomposition, ADMM is another powerful decomposition tool. ADMM is useful for breaking convex optimization problems into small parts to achieve distributed methods, which is elaborated in [39]. In [40], the revenue maximization problem for mobile data offloading in SDN is handled by ADMM. Multi-block ADMM is surveyed in [41] for big data optimization problems in the smart grid. In [42], the authors investigate resource allocation in network virtualization using ADMM in both parallel and distributed fashions. The optimal large-scale scheduling of microgrids using ADMM is proposed in [43].

Although much research has been done in literature, our work is distinctive. For solving the mixed-integer programming problem which is often formulated in the NFV resource allocation, we propose a novel algorithm framework. The framework is composed of Benders decomposition and ADMM. Benders decomposition is designed for decomposing the continuous and integer problem. ADMM is introduced to give a distributed solution for the continuous problem. Both sub-algorithms take advantage of dual information. By adopting the proposed framework, large-scale mixed-integer programming problems can be solved in a distributed manner. As an alternative to the other existing heuristic algorithms studied in the literature, our algorithm has the remarkable performance, effectiveness, and universality.

III. SYSTEM MODEL AND PROBLEM FORMULATION

A. System Model

We consider a virtualization network with N VMs and L links, as shown in Fig. 2. In this paper, we assume that the VMs are already placed on certain physical machines, and the

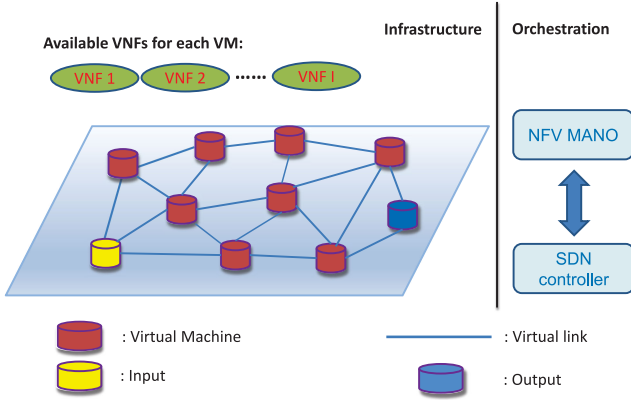


Fig. 2. The model of the network.

input and output nodes are fixed. The VM placement problem can be solved similarly, which is beyond the scope of this paper. Each VM has the capacity to achieve a series of VNFs; however, the capacity is limited. For the n^{th} VM, the capacity is presented as C_n . Moreover, each VNF can be implemented in different VMs at the same time, considering utilizing the resources flexibly. We denote that the VNF set for the network is $\mathcal{F} = \{f_1, \dots, f_I\}$, and the VNF set for each VM is a subset of \mathcal{F} . We also assume that each VM can only process one VNF at the same time. The SDN controller and the NFV MANO are the main central processors, and they have different functions. The SDN controller determines the logic of the traffic and passes the status information to the NFV MANO. The NFV MANO can be divided into three part: the orchestrator, VNFs managers, and virtualized infrastructure managers. The placement of the VNF and the optimization of the traffic path are accomplished by the NFV MANO.

Service function chain (SFC) is defined as a set of VNFs in a certain order to achieve different services. We assume M SFCs are required to traverse through the network, and the VNF set for the m^{th} SFC is expressed as $\mathcal{F}^m = \{f_1^m, \dots, f_K^m | f_k^m \in \mathcal{F}\}$. In addition, each VNF requests a data rate requirement, denoting as R_k . During each SFC period, the network resource allocation orchestration is invariant. For each VNF, the input and output data rates may change because of the specific processing. Thus, we denote the input-output ratio of f_k as μ_{f_k} .

The existence of VNF f_k^m for the n^{th} VM is defined as a binary variable, which can be denote as

$$\delta_{n,k}^m = \begin{cases} 1, & \text{if } f_k^m \text{ is implemented on VM } n, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

For the data flow of the SFC, there may have many paths for one VNF to another. We define the number of the instances of f_k^m is e_k^m , and it can be calculated as

$$e_k^m = \sum_{n=1}^N \delta_{n,k}^m. \quad (2)$$

In other words, e_k^m is a variable depending on $\delta_{n,k}^m$. For the k^{th} VNF of the m^{th} SFC, we define $v_{n,k}^m$ as the traffic volume.

B. Problem Formulation

Our main interest is to minimize the cost of the power consumption for the network by VNF placement and resource allocation. In this paper, we consider the cost can be divided into two part. One is VNF placement and the other is traffic expenditure. We assume that the data flow for each virtual link is lossless. In the previous work of other papers, the power consumption for implementing VNFs have been pointed out. We assume that the cost for each VNF is various, and it is denoted as ξ_k^m for implementing one instance of f_k^m in the k^{th} SFC. Thus, the total cost for VNF placement is formulated as

$$Q_p = \sum_{k=1}^K e_k^m \xi_k^m, \quad (3)$$

Another cost is produced by the traffic for each VNF, and different types of VNF have different cost coefficients, denoting as γ_k for the k^{th} VNF. We assume that the cost is depend linearly on the traffic volume of VNF. Thus, the cost can be formulated as

$$Q_t = \sum_{k=1}^K \gamma_k \sum_{n=1}^N \delta_{n,k}^m v_{n,k}^m. \quad (4)$$

Therefore, the total cost function is

$$Q_{total} = Q_p + Q_t. \quad (5)$$

From above, we can formulate the problem as

$$\min_{\delta_{n,k}^m, v_{n,k}^m} Q_{total} \quad (6)$$

$$\text{s.t } 1 \leq \sum_{n=1}^N \delta_{n,k}^m \leq N \quad \forall k, m, \quad (7)$$

$$\sum_{k=1}^K \delta_{n,k}^m v_{n,k}^m \phi_k \leq C_n \quad \forall n, m, \quad (8)$$

$$\sum_{n=1}^N v_{n,k}^m \geq R_k \quad \forall k, \quad (9)$$

$$\mu_{f_k} \leq \frac{\sum_{n=1}^N v_{n,k+1}^m}{\sum_{n=1}^N v_{n,k}^m} \quad \forall k, \quad (10)$$

$$\delta_{n,k}^m \in \{0, 1\} \quad \forall m, n, k. \quad (11)$$

Here, (7) indicates that all the VNF in the SFC should exist in the network; however, it should not be implemented on every VM. (8) is the constraint for the VM's capacity for holding the VNF set, and ϕ_k is the resource cost for the unit traffic of VNF. (9) is the constraint for the requested traffic of each VNF. (10) indicates the previous VNF output cannot exceed the latter VNF's traffic volume. Finally, (11) is the binary constraint.

Although this problem can be solved in a centralized manner, the computation task for the controller is arduous as an NP-hard problem. The controller must deal with the same sized integer variables and continuous variables. Thus, the decentralized algorithm is necessary. In the following section, we propose an algorithm based on joint Benders decomposition and ADMM

to solve the problem in a distributed manner to release the computational burden of the controller.

IV. ALGORITHM BASED ON BENDERS DECOMPOSITION AND ADMM

In this section, we propose the joint Benders decomposition and ADMM based on the former content. Benders decomposition is popular for separating the mixed integer programming into continuous subproblems and discrete master problem. For the continuous subproblems, we use ADMM to decouple the constraints with associating all variables to generate a distributed solution, which is necessary in the NFV network. However, ADMM is known for solving convex problems distributedly. If the convexity of the problem cannot be satisfied, the convergence and optimal solution are unreliable. Naturally, we use Benders decomposition as the outer loop and use ADMM as the inner loop to solve the subproblem. In the following, we introduce Benders decomposition and ADMM respectively, and show how to integrate them into an algorithm framework. The analysis and implementation are also presented.

A. Benders Decomposition

The problem we formulated is a MILP problem. To solve the problem, we need to deal with the integer variable first. Benders decomposition is an efficient technique for dealing with complicating variables (e.g., integer variables). We treat the Benders decomposition algorithm as the outer loop for our algorithm.

The principle of the Benders decomposition algorithm is separating the continuous variable and integer variable and handling them individually. There are two kinds of problems in the algorithm, which are the master problem and the subproblem. The master problem aims to solve the pure integer programming problem or a small scale mixed integer programming problem. The subproblem is to solve problems only with continuous variables. Benders decomposition is an iteration algorithm. First is the initialization of the problem. Next, the subproblem is solved by involving only continuous variables, because the integer variables are fixed. Through solving the subproblem, the continuous variables solution and the dual variable solution associated with the constraints of fixing the integer variables' value are obtained. After that, the upper and lower bounds are generated using the solutions from solving the subproblem. The difference between the upper and lower bounds is used for the stopping criterion in the iterations. Then, the master problem is solved. The whole procedure of the algorithm to solve the NFV resource allocation problem is stated as follows.

Initialization: Actually, the initialization is generated from the trivial solution of the master problem. First we assign loop counter l to 1. In our problem formulation $\delta_{n,k}^m$ is a binary variable and $U_{\delta_{n,k}^m} = 1$ (upper bound) and $L_{\delta_{n,k}^m} = 0$ (lower bound). Because the coefficient of the corresponding part in the objective function is positive, we simply assign $\delta_{n,k}^{m,(l)} = L_{\delta_{n,k}^m}$. Furthermore, we introduce a function α as an optimal value of the subproblem. The initial value of $\alpha^{(l)}$ is set as α^{down} , which should be determined by the certain circumstances.

Subproblem: The subproblem is constructed by fixing the value of complicating variables to avoid them. Thus, the subproblem can be expressed as the problem (12). $\theta_{n,k}^{m,(l)}$ is the dual variable for the constraints that fixing the binary variables' value. The subproblem is deduced to a problem with only continuous variables and it is often solved in a distributed manner.

$$\min_{\mathbf{v}^m} Q_t \quad (12)$$

$$\text{s.t.} \quad \sum_{k=1}^K \delta_{n,k}^m v_{n,k}^m \phi_k \leq C_n \quad \forall n, m, \quad (13)$$

$$\sum_{n=1}^N v_{n,k}^m \geq R_k \quad \forall k, \quad (14)$$

$$\mu_{f_k} \leq \frac{\sum_{n=1}^N v_{n,k+1}^m}{\sum_{n=1}^N v_{n,k}^m} \quad \forall k, \quad (15)$$

$$\delta_{n,k}^m = \delta_{n,k}^{m,(l)} : \theta_{n,k}^{m,(l)} \quad \forall m, n, k. \quad (16)$$

In this paper, we adopt ADMM to solve the subproblem which is illustrated in the next section. By solving the subproblem, we can acquire $\mathbf{v}^{m,(l)}$ and $\theta^{m,(l)}$ for the following steps.

Convergence checking: The convergence checking is significant for the algorithm because it performs as the stopping criterion. The upper bound for the primal problem is calculated as

$$Q_{up}^l = Q_{total}^l(\delta^{m,(l)}, \mathbf{v}^{m,(l)}). \quad (17)$$

Correspondingly, the lower bound can be generated by

$$Q_{down}^l = Q_p^l(\delta^{m,(l)}) + \alpha^{(l)}. \quad (18)$$

Thus, the stopping criterion is given by

$$\begin{cases} Q_{up}^l - Q_{down}^l \leq \varepsilon, & \text{stop,} \\ \text{otherwise,} & \text{continue,} \end{cases} \quad (19)$$

where ε is a pre-defined tolerance for the problem. Once the iteration stops, the optimal solution is obtained as $\delta^{m,(l)}$ and $\mathbf{v}^{m,(l)}$.

Master problem: The master problem is only related to the binary variables. After update the loop counter $l = l + 1$, the problem which need to solve is stated as follows

$$\min_{\delta^{m,\alpha}} Q_p + \alpha \quad (20)$$

$$\text{s.t.} \quad 1 \leq \sum_{n=1}^N \delta_{n,k}^m \leq N \quad \forall k, m, \quad (21)$$

$$Q_t(\mathbf{v}^{m,(j)}) + \theta^{m,(j)} (\delta^m - \delta^{m,(j)}) \leq \alpha \quad j = 1, \dots, l-1, \quad (22)$$

$$\sum_{k=1}^K \delta_{n,k}^m v_{n,k}^m \phi_k \leq C_n \quad \forall n, m, \quad (23)$$

$$\alpha \geq \alpha^{down}, \quad (24)$$

$$\delta_{n,k}^m \in \{0, 1\} \quad \forall m, n, k, \quad (25)$$

Algorithm 1: NFV Resource Allocation Based on Benders Decomposition.

```

1: Initialize: loop index  $l$ ,  $U_{\delta_{n,k}^m}$ ,  $L_{\delta_{n,k}^m}$ ,  $\alpha^{(l)}$ 
2: while  $Q_{up}^l - Q_{down}^l \geq \varepsilon$  do
3:   Subproblem
4:   acquire  $\mathbf{v}^{m(l)}$  and  $\theta^{m,(l)}$  using ADMM
5:   Bounds calculation
6:   calculate upper and lower bound ( $Q_{up}^l$  and  $Q_{down}^l$ )
7:   by (17) and (18)
8:   Master problem
9:   step 1: update loop index  $l = l + 1$ 
10:  step 2: add new Benders cut to the problem (20)
11:  step 3: solve the problem in (20) to obtain the
12:  optimal value of  $\delta^m$  and  $\alpha$ 
13: end while

```

where the constraint (22) is the Benders cuts associated with the former iterations. In every iteration, the new benders cut will be added to the constraints of the master problem. The Benders cuts compose $l - 1$ hyperplanes to approximate the objective function of the subproblem from below.

After the master problem is solved, the algorithm goes to the next iteration with solving the subproblem. The whole procedure of the algorithm is presented in the Algorithm 1.

B. ADMM

For our designed continuous subproblem, ADMM is the most suitable algorithm. For example, dual decomposition is an efficient way to generate a distributed solution. However, it cannot deal with the constraints coupling with different users. OCD (Optimal Condition Decomposition) is another popular method in distributed computing. Unfortunately, our problem cannot satisfy the strong convexity condition (linear programming), which will cause convergence problem. Many other distributed computing methods have strict requirements which are inappropriate for our subproblem. Thus, we adopt ADMM for solving the subproblem in a distributed manner. ADMM is a method to solve the problem by decoupling the constraints and breaking it into many small problems. For the problem with a general form stated as follows

$$\begin{aligned} \min \quad & f(x) + g(z) \\ \text{s.t.} \quad & Ax + Bz = c. \end{aligned} \quad (26)$$

The optimization problem has variables x and z with the equality constraint. The augmented Lagrangian function for the problem is expressed as

$$\begin{aligned} \mathcal{L}(x, z, \lambda) = & f(x) + g(z) + \lambda^T (Ax + Bz - c) \\ & + \frac{\rho}{2} \|Ax + Bz - c\|^2. \end{aligned} \quad (27)$$

The Lagrangian multiplier is denoting as λ and ρ is the penalty parameter. The ADMM iterations proceed as follows

sequentially

$$x[t + 1] := \arg \min_x \mathcal{L}(x, z[t], \lambda[t]), \quad (28)$$

$$z[t + 1] := \arg \min_z \mathcal{L}(x[t + 1], z, \lambda[t]), \quad (29)$$

$$\lambda[t + 1] := \lambda[t] + \rho (Ax[t + 1] + Bz[t + 1] - c). \quad (30)$$

The iterations stop when a stopping criterion meets.

For the proposed subproblem, we need to transform problem (12)–(16) into an equivalent problem. Among the constraints of the problem, we can find that (14) and (15) are constraints that involve all the VMs. To generate a distributed algorithm, we introduce an auxiliary variable as

$$v_{n,k}^m = u_{n,k}^m \quad \forall n, k, \quad (31)$$

where $u_{n,k}^m$ is a copy of the $v_{n,k}^m$. Thus, the constraints in (9) and (10) can be reformulated as

$$\sum_{n=1}^N u_{n,k}^m \geq R_k \quad \forall k, \quad (32)$$

$$\mu_{f_k} \sum_{n=1}^N u_{n,k}^m - \sum_{n=1}^N u_{n,k+1}^m \leq 0 \quad \forall k. \quad (33)$$

Thus, the equivalent problem is formulated as

$$\min_{\mathbf{v}^m} Q_t \quad (34)$$

$$\text{s.t.} \quad \sum_{k=1}^K \delta_{n,k}^m v_{n,k}^m \phi_k \leq C_n \quad \forall n, m, \quad (35)$$

$$\sum_{n=1}^N u_{n,k}^m \geq R_k \quad \forall k, \quad (36)$$

$$\mu_{f_k} \sum_{n=1}^N u_{n,k}^m - \sum_{n=1}^N u_{n,k+1}^m \leq 0 \quad \forall k \quad (37)$$

$$\delta_{n,k}^m = \delta_{n,k}^{m,(l)} : \theta_{n,k}^{m,(l)} \quad \forall m, n, k \quad (38)$$

$$v_{n,k}^m = u_{n,k}^m \quad \forall n, k. \quad (39)$$

The augmented Lagrangian function of the problem is given by

$$\begin{aligned} \mathcal{L}_\rho = & Q_t + \sum_{n=1}^N \sum_{k=1}^K \lambda_{n,k} (v_{n,k}^m - u_{n,k}^m) \\ & + \frac{\rho}{2} \sum_{k=1}^K \sum_{n=1}^N \|v_{n,k}^m - u_{n,k}^m\|_2^2, \end{aligned} \quad (40)$$

where λ_n is the Lagrangian multiplier and $\rho > 0$ is the penalty parameter. For the t^{th} iteration, the primal and dual variables update as follow

$$\mathbf{u}[t + 1] = \arg \min \mathcal{L}_\rho(\mathbf{u}, \mathbf{v}[t], \boldsymbol{\lambda}[t]), \quad (41)$$

$$\mathbf{v}[t + 1] = \arg \min \mathcal{L}_\rho(\mathbf{u}[t + 1], \mathbf{v}, \boldsymbol{\lambda}[t]), \quad (42)$$

$$\boldsymbol{\lambda}[t + 1] = \boldsymbol{\lambda}[t] + \rho(\mathbf{u}[t + 1] - \mathbf{v}[t + 1]). \quad (43)$$

Thus, the variables can update sequentially in iterations.

From the formulation of the problem, we can compress and separate the problem for simplicity. We deem that the variable $u_{n,k}^m$ is a global copy of $v_{n,k}^m$, which means that it is handled by the controller. The problem for updating $u_{n,k}^m$ is given by

$$\begin{aligned} \min_{\mathbf{u}^{m,(t)}} \quad & \sum_{n=1}^N \sum_{k=1}^K -\lambda_{n,k} u_{n,k}^m + \frac{\rho}{2} \sum_{n=1}^N \sum_{k=1}^K \|\tilde{v}_{n,k}^m - u_{n,k}^m\|_2^2 \\ \text{s.t.} \quad & \sum_{n=1}^N u_{n,k}^m \geq R_k \quad \forall k, \\ & \mu f_k \sum_{n=1}^N u_{n,k}^m - \sum_{n=1}^N u_{n,k+1}^m \leq 0 \quad \forall k, \end{aligned} \quad (44)$$

where $\tilde{v}_{n,k}^m$ is a constant passed by the former iteration. For $v_{n,k}^m$, it is considered as the local variable for every VM. Therefore, the problem for each VM of updating $v_{n,k}^m$ is given by

$$\begin{aligned} \min_{\mathbf{v}^{m,(t)}} \quad & \sum_{k=1}^K \gamma_k v_{n,k}^m + \sum_{k=1}^K \lambda_{n,k} + \frac{\rho}{2} \sum_{k=1}^K \|v_{n,k}^m - \tilde{u}_{n,k}^m\|_2^2 \\ \text{s.t.} \quad & \sum_{k=1}^K \delta_{n,k}^m v_{n,k}^m \phi_k \leq C_n \quad \forall m. \end{aligned} \quad (45)$$

In addition, the dual variable $\lambda_{n,k}$ is updated at each VM simply as

$$\lambda_{n,k}[t+1] = \lambda_{n,k}[t] + \rho(\tilde{u}_{n,k}^m - \tilde{v}_{n,k}^m) \quad \forall n, k, m. \quad (46)$$

Therefore, the whole procedure of ADMM is described in Algorithm 2.

C. Analysis

The outer loop is based on Benders decomposition, which is responsible for dealing with binary variables. The convergence of the Benders decomposition is guaranteed if the α is a convex function of δ^m . Thus, the theorem is proposed as follows.

Theorem 1: α is a convex function of δ^m , which ensures the outer loop convergence.

The proof of Theorem 1 is illustrated in Appendix. The subproblem is designed as an inner loop iteration based on ADMM. The convergence of the ADMM is illustrated in the following Remark 1.

Remark 1: For solving a convex optimization problem, ADMM guarantees to convergence [39]. For the formulated problem, the objective function of the subproblem is closed, proper and convex. Moreover, the constraints are linear, and the strong duality holds for such linear programming. Thus, ADMM for our problem can converge definitely.

Benders decomposition reduces the complexity of solving the original MILP by decomposing the problem into a series of independent smaller subproblems [44]. In each iteration, the Benders cuts eliminate feasible regions without optimal solution [45]. For ADMM adopted in subproblem, it will return ϵ -optimal solution within $\mathcal{O}(1/\epsilon^2)$ iterations [46].

The large-scale continuous variables are decoupled for each VM. Although our algorithm is not in a fully distributed manner,

Algorithm 2: Distributed Algorithm for the Subproblem Based on ADMM.

```

1: Initialize:  $t, \lambda, \mathbf{v}$ 
2: while the stopping criterion is not satisfied do
3:   Controller update
4:   repeat
5:     wait
6:   until receive updated  $\lambda, \mathbf{v}$  from all  $N$  distributed VMs
7:     step 1: Solve the problem in (44) and obtain the
8:       optimal solution  $\tilde{\mathbf{u}}$ 
9:     step 2: Then, send  $\tilde{\mathbf{u}}$  to the VMs
10:    step 3: Update  $t = t + 1$ 
11:
12:   Each VM updates
13:   repeat
14:     wait
15:   until receive updated  $\tilde{\mathbf{u}}$  from the controller
16:     step 1: Solve the problem in (45) and obtain the
17:       optimal solution  $\tilde{\mathbf{v}}$ 
18:     step 2: Update dual variables:
19:        $\lambda[t+1] = \lambda[t] + \rho(\tilde{\mathbf{u}}[t+1] - \tilde{\mathbf{v}}[t+1])$ 
20:     step 3: Send the updated  $\tilde{\mathbf{v}}$  and  $\lambda[t+1]$  back to the
21:       controller for the next iteration
22:   end while

```

there are still many benefits. Firstly, each VM node can update variables without information exchange with other VM nodes which will reduce the cost of signaling. Then, the heavy tasks are allocated to the central controller, which can take full advantage of the calculation ability of the network. It is possible that the subproblem sometimes encounters infeasibility. The method to handle the infeasibility is adding artificial variables, which is explained in [35] in details. However, the cost for this method is introducing a more substantial number of variables which will increase the problem complexity.

D. Algorithm Implementation

The whole procedure of the algorithm is depicted in Fig. 3. Benders decomposition is famous for separating the mixed-integer problems into continuous and discrete problems. ADMM is widely used for decouple the complicated constraints and give a distributed solution. Both of the algorithms are iterative algorithms. Moreover, ADMM is introduced to solve only the subproblem of the Benders decomposition. Thus, the continuous subproblem should have variables coupled constraints for ADMM. The stopping criterions need to be chosen properly according to [47] and [48]. If the threshold is picked too small, it may cost too much time for ADMM to converge. However, if the threshold value is too large, the rough solution may cause the divergence of the algorithm. For the subproblem, the optimization task is done by both controller and VMs using ADMM. The controller is responsible for updating the global copy variable denoting as \mathbf{u} and sending them to VMs after the

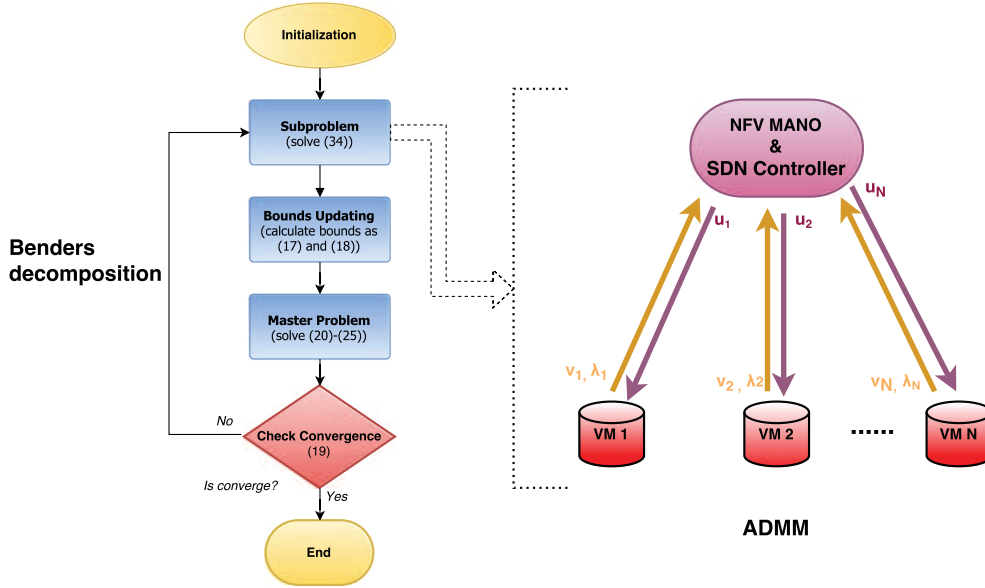


Fig. 3. The illustration of the Benders decomposition algorithm procedure and the information exchange in subproblem using ADMM.

calculation. Each VM updates its own variables including the traffic variable and dual variable. Then, they send their updating results back to the controller for the next iteration. This procedure is repeated until convergence. Note that, the subproblem is decoupled and VMs can solve the optimization problem by themselves without knowing information from other VMs. The bounds updating is calculated at the controller. For the master problem, the controller solve a small scale MILP comparing to the original MILP. After each iteration, more Benders cuts will be added to the problem to cut the feasible region and make it smaller and smaller. When the difference between the upper and lower bounds calculated at the controller is smaller than a certain threshold, the optimal resource allocation strategy is obtained. In the vehicular network, the high dynamic scenarios require the mobility of the system. If the system need to be operated at different positions or scenarios, the proposed algorithm framework is still feasible for the most of the cases. Difference of the parameters will not affect the feasibility of the designed architecture. The proposed architecture of NFV is well fitted in the vehicular network. As the service condition is changing, the controller can make efficient computation strategy for the network with this architecture.

V. IMPLEMENTATION USING HADOOP MAPREDUCE

In this section, we introduce an overview of MapReduce programming model and describe the detailed implementation of ADMM-based Benders decomposition in Algorithm 1 using the Hadoop MapReduce framework.

A. MapReduce Programming Model

MapReduce is a programming model for distributed processing of extensive datasets using a large cluster of commodity machines [49]. It has been widely used to perform special-purpose computations both in industry and academia [50]. A

MapReduce computation consists of a set of Map tasks and Reduce tasks. The input data will be split into independent blocks and processed by the Map tasks in a completely parallel manner to produce a set of intermediate key-value pairs. Then, all outputs of the mapping operation that share the same intermediate key will be grouped and passed to the same Reducer. Generally, the Map and Reduce steps can be conceptually expressed as [49]

$$\begin{aligned} \text{map} \quad (k_1, v_1) &\quad \longrightarrow \text{list}(k_2, v_2) \\ \text{reduce} \quad (k_2, \text{list}(v_2)) &\quad \longrightarrow \text{list}(v_3). \end{aligned}$$

Apache Hadoop is an open-source software framework written in Java for easily writing the application to process the massive amount of data on computer clusters in reliable, fault-tolerant manner [51]. The core of Hadoop consists of a storage part, which provides the Hadoop Distributed File System (HDFS) architecture, and a processing part which implements the MapReduce computation paradigm. The HDFS manages the storage of data across an entire cluster of machines by splitting files into blocks and distributing them amongst the nodes in the cluster [52]. Then, the data at each node is divided into the fixed-size piece called splits. Each split of data is processed in the Map tasks based on the user-defined Map function to produce a list of key-value pairs. The process of sorting key-value pairs of map tasks and sending them to reducers is handling internally by Hadoop. This allows Hadoop to reduce many complexities such as data partitioning, scheduling tasks across many machines, handling machine failures and performing inter-machine communication [53].

B. Implementation Using Hadoop MapReduce

Each iteration of ADMM in Algorithm 2 can be represented as a MapReduce job as illustrated in Fig. 4. The distributed computations for each VM's subproblems in (45) are performed by Map tasks, and the central controller subproblem

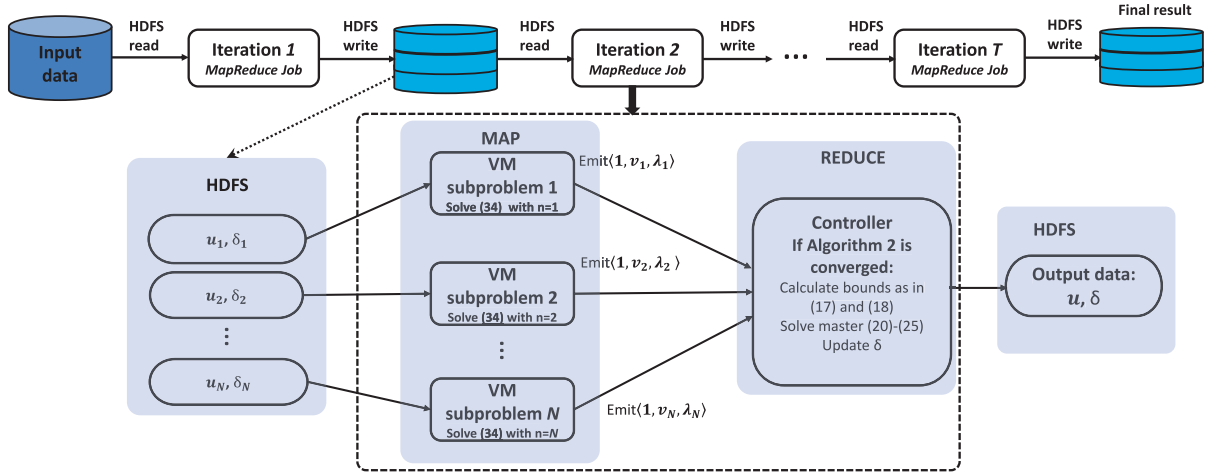


Fig. 4. Data sharing for iterative ADMM using Hadoop MapReduce and the detailed illustration of Map tasks and Reduce task in each MapReduce job in each iteration.

computation in (44) and master problem in (20) are performed by a Reduce task. We have total N Mappers, one for each VM's subproblem. Each Mapper solves the optimization problem in (44) to obtain \mathbf{v}_n . However, solving the problem in (45) for $\mathbf{v}_n[t+1]$ on iteration $t+1$ needs to use $\mathbf{u}_n[t]$ and $\boldsymbol{\lambda}_n[t]$ from the previous iteration. Since MapReduce is not designed to support iterative applications, we facilitate iterative computation for the Joint Bender decomposition and ADMM in Algorithm 1 by writing the output data at each iteration to the HDFS, which will be used as the input data for Mappers in the next iteration. Particularly, each Mapper uses vm_ID provided by Hadoop to identify which islanded problem is and loads the corresponding $\boldsymbol{\lambda}_n[t], \mathbf{u}_n[t], \boldsymbol{\delta}[t]$ from HDFS in the previous iteration.

After solving the optimization problem, each Mapper updates values for $\boldsymbol{\lambda}_n$ using (46). Then, each Mapper emits an intermediate key-value pair, which is $\langle 1, \{\mathbf{v}_n, \boldsymbol{\lambda}_n\} \rangle$ to the Reducer. Since in our problem, there is a single Reducer, which plays the role of performing the central controller subproblem and master problem, all the keys in all Map tasks are selected as 1 to force all information from the Mappers is sent to a unique Reducer. Based on all information received from the Mappers, the Reducer solves (44) to obtain $\{\mathbf{u}_n\}_{\forall n}$. Then, the Reducer checks if Algorithm 2 is converged, it calculates the upper bound, lower bound, and solves the master optimization problem in (20).

The values of $\{\mathbf{u}_n\}_{\forall n}, \boldsymbol{\lambda}, \boldsymbol{\delta}$ are written out to HDFS directly by the Reducer, which will be used as the input data for MapReduce job in the next iteration. The detailed Map tasks and Reduce task in each iteration is illustrated in Fig. 4. The pseudo-code for implementing the Joint Benders decomposition and ADMM Algorithm 1 using Hadoop MapReduce is described in Algorithm 3.

In order to implement our algorithm using Hadoop MapReduce, we set up a cluster with 4 computers. Each computer has an Intel Core2 Quad CPU Q8200, a 4 GB memory, and a 64-bit ubuntu 16.04 OS. In our Java code, we use Gurobi solver to solve the problems in both Mappers and Reducers. The running time is shown in Fig. 5. We run over 100 times and take the average result. From the figure, we can notice that the running time tends

Algorithm 3: ADMM-Based Bender Decomposition Using Hadoop MapReduce.

```

1: function MAP(vm_ID, inputData)
2:   Load data of previous iteration from HDFS
3:   corresponding to vm_ID
4:   Solve subproblem corresponding to each VM in
5:   (45)
6:   Update  $\boldsymbol{\lambda}_n$  using (46)
7:   EMIT  $\langle 1, \{\mathbf{v}_n, \boldsymbol{\lambda}_n\} \rangle$ 
8: end Function
9: function REDUCE(key, Data from Mappers)
10:  Concatenate  $\{\mathbf{v}_n, \boldsymbol{\lambda}_n\}$  from all VMs
11:  Solve controller subproblem in (44) for  $\mathbf{u}$ 
12:  if Algorithm 2 is converged then
13:    Calculate Upper bound and Lower bound as in
14:    (17)
15:    and (18)
16:    Solve master problem in (20)
17:    Update  $\boldsymbol{\delta}$ 
18:  end if
19:  EMIT  $\langle \{\mathbf{u}, \boldsymbol{\delta}\} \rangle$ 
20: end Function
21: function MAIN(inputPath, outputPath)
22:  Initialization
23:  while  $Q_{up}^l - Q_{down}^l \geq \epsilon$  do
24:    run MapReduceJob (inputPath, outputPath)
25:     $t \leftarrow t + 1$ 
26:  end while
27: end Function

```

to increase with the increase of the number of VM. However, the increasing rate is slow. This is because the major time is caused by write/read data from HDFS and subproblems on different Mappers. If the problem size keeps increasing, the cluster will cost more time to converge. The results show that our algorithm

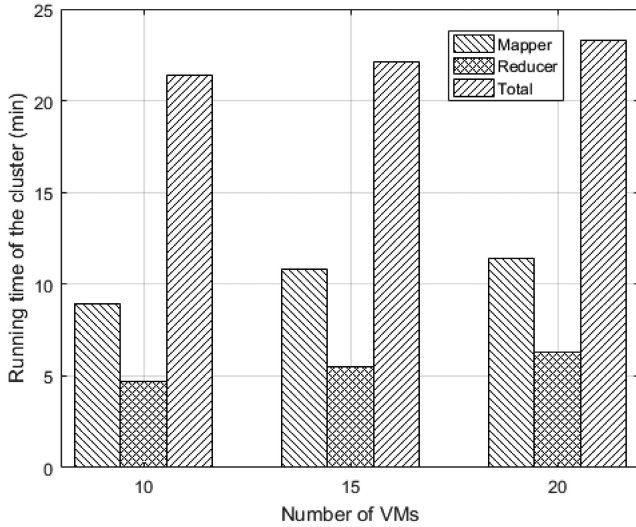


Fig. 5. The running time on the cluster.

 TABLE II
 PARAMETERS FOR VNFs

	VNF1	VNF2	VNF3	VNF4	VNF5
Implementation cost ξ_k^m	0.2772	0.4596	0.2415	0.1481	0.3276
Scaling ratio μ_{f_k}	0.6125	0.7440	0.5293	0.5996	0.5536
Traffic cost γ_k	uniformly distributed in the interval [1,2]				
Resource cost ϕ_k	uniformly distributed in the interval [1,10]				

can be implemented on Hadoop MapReduce, which will benefit us for processing data when the incoming data amount is large.

VI. SIMULATION RESULTS

In this section, we present the simulation results to evaluate the effectiveness of our proposed algorithm. For the service function chain, we assume that the function number $K = 5$, which are randomly picked among firewall, proxy, network address translation (NAT) and intrusion detection systems (IDSs), and so on. The corresponding function parameter is stated in Table II, similar to the set up in [9]. The data rate for the service chaining is chosen from the set as {6 8 10 12 14} MB/s according to a Zipf distribution [54]. For our evaluation, we consider a fully connected network. We set the number of VMs N as 10. For every VM, the capacity is set as a uniformly distributed random number from 1 to 10 [9]. Resources for every VM (CPU, Memory, Disk) are assumed as the same kind of resource. The lower bound α^{down} for α is initialized as a small enough number, according to [35]. We assume that each VNF can at least be processed by one VM. All the simulation results are run in MATLAB R2016b. All optimization problems are solved by using CVX. Although the simulation is far from considering all cases in practical networking, the results can give an overview of the effectiveness of our proposed algorithm.

A. Convergence

Fig. 7 and Fig. 6 show the convergence of the proposed algorithms. Fig. 7 presents the average convergence performance

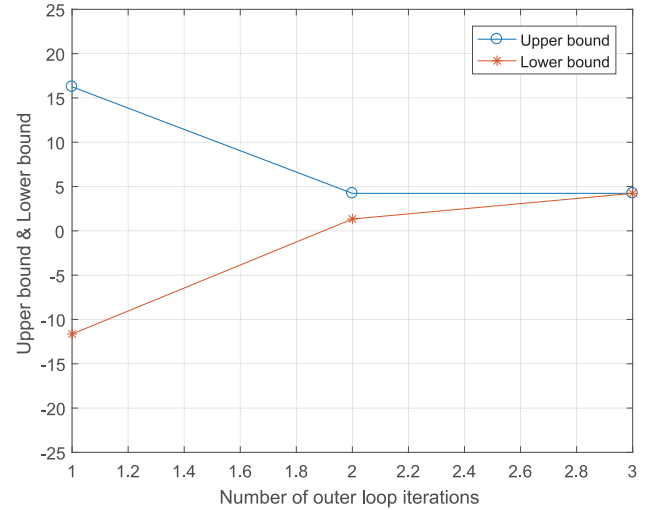


Fig. 6. The convergence performance of Benders decomposition.

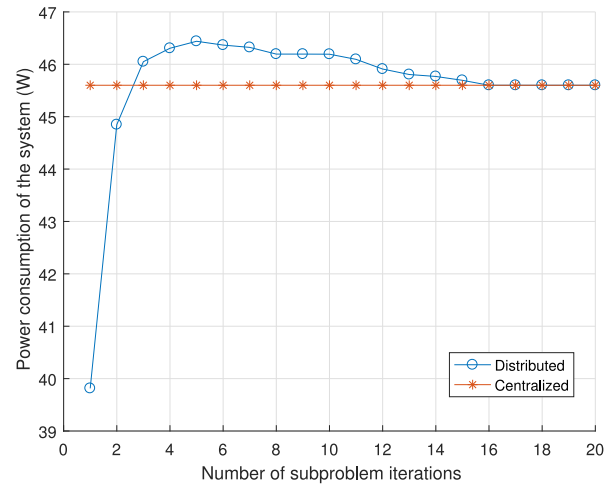


Fig. 7. The convergence performance of ADMM.

of ADMM for solving the subproblem. To verify the convergence, we deploy both centralized and distributed algorithms. The centralized algorithm is deemed as a baseline, which is optimizing subproblem without decomposition. As shown in the figure, the distributed algorithm converges to the same optimal value as the centralized algorithm within several iterations.

In Fig. 6, the average convergence of Benders decomposition is presented. From the figure, we can see that after 3 outer loop iterations, the algorithm converges. The difference between the upper bound and lower bound is decreasing, and then finally converges. One thing needs to be noticed is that the upper bound does not need to be monotonous, but the lower bound is monotonously increasing with the iterations all the time [35].

B. Computational Performance

In Fig. 8, it demonstrates that the relationship between the average execution time and the number of VMs. When the number of VMs is increasing from 10 to 80, the average master problem execution time is increasing. This is due to the fact that

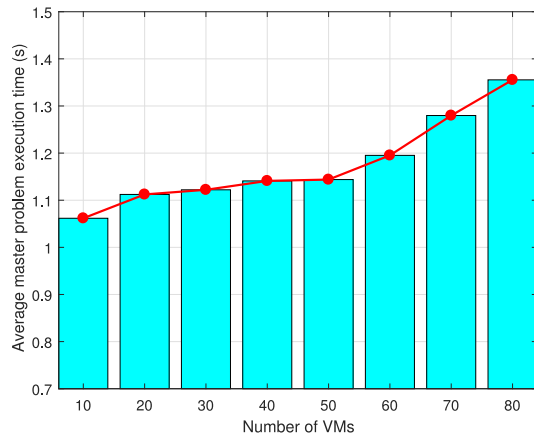


Fig. 8. The average execution time of master problem versus the number of VMs.

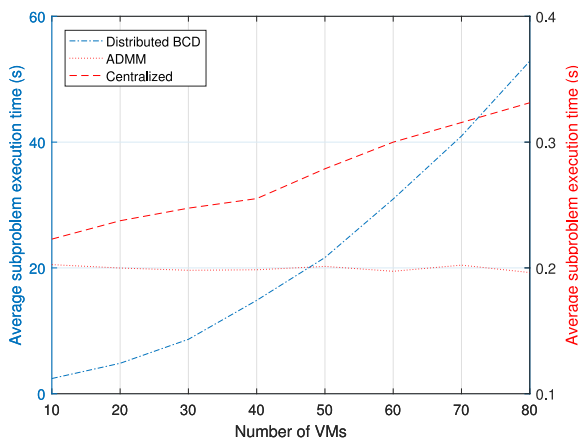


Fig. 9. The average execution time of subproblem versus the number of VMs.

more VMs will make the variables scale up. Thus, the time for solving mater problem will increase.

In Fig. 9, the average execution time of the subproblem with different numbers of VMs is presented, including both distributed BCD, ADMM, and centralized algorithm. For the distributed BCD algorithm, the execution time increasing fast when the number of VM is increasing. Comparing to ADMM, the distributed BCD solves the supproblem sequentially. Thus, if the number of VM increases, there will be more tasks for distributed BCD to execute. When the number of VMs is increasing, the time for the centralized algorithm is increasing while the distributed algorithm almost remains unchanged. For the centralized manner, the controller needs to solve the large-scale problem to obtain the optimal solutions for all the VMs. With the increase of VMs, the scaled-up problem needs more time to handle. However, using ADMM as a distributed algorithm, the problem for every VM is still the same size when the number of VMs increases. Thus, the computational time is greatly reduced adopting ADMM.

C. System Performance

In Fig. 10, the total system cost versus request data rate of SFC is presented. Note that, since the outer loop is terminated

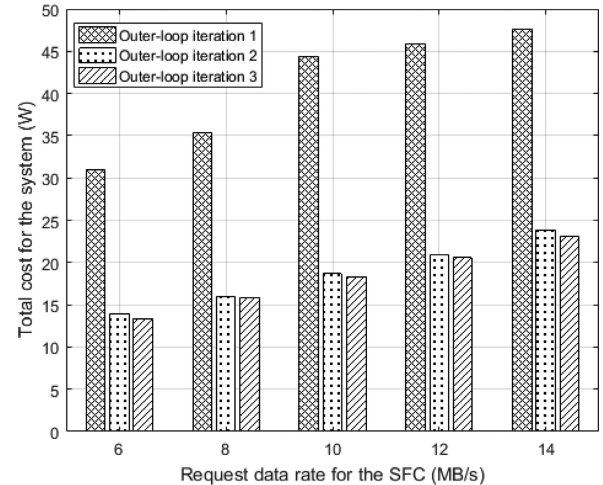


Fig. 10. The total system cost versus request data rate.

after three iterations, we give three results for the iterations and the outer-loop iteration 3 is optimal for the optimization. Furthermore, it is shown that when the request data for the SFC increases, the total cost for the system increases. This is due to more traffic volume will bring more costs for the network. After the first iteration, the total cost reduced a lot. The difference of the optimal value between the last two iterations is small.

VII. CONCLUSION

In this paper, we first propose an algorithm framework based on joint Benders decomposition and ADMM to solve the resource allocation in an NFV network. The formulated resource allocation problem, which involves integer and continuous variables, is composed by both VNF placement problem and traffic allocation problem. To solve this mixed integer problem, we design Benders decomposition as the outer loop algorithm to separate the integer variables and continuous variables. The subproblem is continuous and the master problem is discrete. The subproblem with a large-scale of continuous variables is divided into small problems, which are solved distributedly using ADMM. This semi-distributed architecture can release the computation burden of the controller, and increase the flexibility of the system simultaneously. After a finite number of iterations, the optimal solution can be obtained. Then, the implementation of the algorithm for parallel computing using the Hadoop MapReduce software framework is presented. Finally, the simulation results indicate that our proposed algorithm has satisfied convergence and performance.

APPENDIX

PROOF OF BENDERS DECOMPOSITION CONVERGENCE

The convergence of Benders decomposition is determined by the convexity of α . Thus, we only need to prove the α is a convex function of δ^m .

First of all, it is obvious that constraints for (6) construct a convex polytope, and the constraints for the subproblem is a subset of (6). We assume that the two feasible solution sets for the problem in (6) are $\delta^{m,(1)}, \mathbf{v}^{m,(1)}$ and $\delta^{m,(2)}, \mathbf{v}^{m,(2)}$. The

solutions are associated by

$$\alpha(\delta^{m,(1)}) = Q_t(\mathbf{v}^{m,(1)}), \quad (47)$$

$$\alpha(\delta^{m,(2)}) = Q_t(\mathbf{v}^{m,(2)}). \quad (48)$$

Thus, the linear combination of the solutions can be expressed as

$$\delta^{m,(3)} = \beta\delta^{m,(1)} + (1 - \beta)\delta^{m,(2)}, \quad (49)$$

$$\mathbf{v}^{m,(3)} = \beta\mathbf{v}^{m,(1)} + (1 - \beta)\mathbf{v}^{m,(2)}. \quad (50)$$

The objective function of the subproblem at $\mathbf{v}^{m,(3)}$ is given by

$$Q_t(\mathbf{v}^{m,(3)}) = \sum_{k=1}^K \gamma_k \sum_{n=1}^N v_{n,k}^{m,(3)} \quad (51)$$

$$= \sum_{k=1}^K \gamma_k \sum_{n=1}^N (\beta v_{n,k}^{m,(1)} + (1 - \beta)v_{n,k}^{m,(2)}) \quad (52)$$

$$= \beta \sum_{k=1}^K \gamma_k \sum_{n=1}^N v_{n,k}^{m,(1)} + (1 - \beta) \sum_{k=1}^K \gamma_k \sum_{n=1}^N v_{n,k}^{m,(2)} \quad (53)$$

$$= \beta\alpha(\delta^{m,(1)}) + (1 - \beta)\alpha(\delta^{m,(2)}). \quad (54)$$

When the binary variable is fixed as $\delta^{m,(3)}$, the subproblem's optimal solution $\mathbf{v}^{m,*}$ can be obtained. So we have the following conclusion

$$Q_t(\mathbf{v}^{m,*}) \leq Q_t(\mathbf{v}^{m,(3)}). \quad (55)$$

Thus, we proved the convexity of α with variables δ^m by the inequality as follows

$$\alpha(\delta^{m,(3)}) \leq \beta\alpha(\delta^{m,(1)}) + (1 - \beta)\alpha(\delta^{m,(2)}). \quad (56)$$

REFERENCES

- [1] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Serv. Manage.*, vol. 13, no. 3, pp. 518–532, Sep. 2016.
- [2] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.
- [3] R. H. Gau, H. T. Chiu, and P. K. Tsai, "Optimizing the service capacity of SDN-based cellular networks with service chaining and NFV," in *Proc. IEEE 27th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun.*, Valencia, Spain, Sep. 2016, pp. 1–6.
- [4] S. D'Oro, L. Galluccio, S. Palazzo, and G. Schembra, "Exploiting congestion games to achieve distributed service chaining in NFV networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 2, pp. 407–420, Feb. 2017.
- [5] K. Yang, S. Martin, C. Xing, J. Wu, and R. Fan, "Energy-efficient power control for device-to-device communications," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3208–3220, Dec. 2016.
- [6] J. An, Y. Zhang, X. Gao, and K. Yang, "Energy-efficient base station association and beamforming for multi-cell multiuser systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2690–2702, Apr. 2018.
- [7] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
- [8] R. Wang and V. K. N. Lau, "Delay-aware two-hop cooperative relay communications via approximate MDP and stochastic learning," *IEEE Trans. Inf. Theory*, vol. 59, no. 11, pp. 7645–7670, Nov. 2013.
- [9] L. Wang, Z. Lu, X. Wen, R. Knopp, and R. Gupta, "Joint optimization of service function chaining and resource allocation in network function virtualization," *IEEE Access*, vol. 4, pp. 8084–8094, 2016.
- [10] T. Park, Y. Kim, J. Park, H. Suh, B. Hong, and S. Shin, "QoSE: Quality of security a network security framework with distributed NFV," in *Proc. IEEE Int. Conf. Commun.*, Kuala Lumpur, Malaysia, May 2016, pp. 1–6.
- [11] N. M. M. K. Chowdhury and R. Boutaba, "Network virtualization: State of the art and research challenges," *IEEE Commun. Mag.*, vol. 47, no. 7, pp. 20–26, Jul. 2009.
- [12] G. Casale, S. Kraft, and D. Krishnamurthy, "A model of storage I/O performance interference in virtualized systems," in *Proc. 31st Int. Conf. Distrib. Comput. Syst. Workshops*, Jun. 2011, pp. 34–39.
- [13] V. Sciancalepore, F. Giust, K. Samdanis, and Z. Youasaf, "A double-tier MEC-NFV architecture: Design and optimisation," in *Proc. IEEE Conf. Standards Commun. Netw.*, Berlin, Germany, Oct. 2016, pp. 1–6.
- [14] H. Wu, A. N. Tantawi, Y. Diaa, and W. Wang, "Adaptive memory load management in cloud data centers," *IBM J. Res. Develop.*, vol. 55, no. 6, pp. 5:1–5:10, Nov. 2011.
- [15] A. Marotta and A. Kassler, "A power efficient and robust virtual network functions placement problem," in *Proc. 28th Int. Teletraffic Congr.*, Würzburg, Germany, Sep. 2016, vol. 1, pp. 331–339.
- [16] M. Shifrin, E. Biton, and O. Gurewitz, "Optimal control of VNF deployment and scheduling," in *Proc. IEEE Int. Conf. Sci. Elect. Eng.*, Eilat, Israel, Nov. 2016, pp. 1–5.
- [17] W. Xie, J. Zhu, C. Huang, M. Luo, and W. Chou, "Network virtualization with dynamic resource pooling and trading mechanism," in *Proc. IEEE Global Commun. Conf.*, Austin, TX, USA, Dec. 2014, pp. 1829–1835.
- [18] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: A survey," *IEEE Commun. Mag.*, vol. 51, no. 11, pp. 24–31, Nov. 2013.
- [19] I. Volvach and L. Globa, "Mobile networks disaster recovery using SDN-NFV," in *Proc. Int. Conf. Radio Electron. Info Commun.*, Kiev, Ukraine, Sep. 2016, pp. 1–3.
- [20] N. Herbaut, D. Negru, D. Magoni, and P. A. Frangoudis, "Deploying a content delivery service function chain on an SDN-NFV operator infrastructure," in *Proc. Int. Conf. Telecommun. Multimedia*, Heraklion, Greece, Jul. 2016, pp. 1–7.
- [21] H. A. Akyildiz and E. Saygun, "SDN-NFV-cloud introduction in the context of service chaining," in *Proc. 23rd Signal Process. Commun. Appl. Conf.*, Malatya, Turkey, May 2015, pp. 2605–2608.
- [22] Y. Nam, S. Song, and J. M. Chung, "Clustered NFV service chaining optimization in mobile edge clouds," *IEEE Commun. Lett.*, vol. 21, no. 2, pp. 350–353, Feb. 2017.
- [23] H. Moens and F. D. Turck, "Customizable function chains: Managing service chain variability in hybrid NFV networks," *IEEE Trans. Netw. Serv. Manage.*, vol. 13, no. 4, pp. 711–724, Dec. 2016.
- [24] J. Wang, B. He, J. Wang, and T. Li, "Intelligent VNFs selection based on traffic identification in vehicular cloud networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4140–4147, Nov. 2018.
- [25] M. Zhu, J. Cao, Z. Cai, Z. He, and M. Xu, "Providing flexible services for heterogeneous vehicles: An NFV-based approach," *IEEE Netw.*, vol. 30, no. 3, pp. 64–71, May 2016.
- [26] J. Prados-Garzon, J. J. Ramos-Munoz, P. Ameigeiras, P. Andres-Maldonado, and J. M. Lopez-Soler, "Modeling and dimensioning of a virtualized MME for 5G mobile networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 5, pp. 4383–4395, May 2017.
- [27] P. Demestichas, A. Georgakopoulos, K. Tsagkaris, and S. Kotrotsos, "Intelligent 5G networks: Managing 5G wireless mobile broadband," *IEEE Veh. Technol. Mag.*, vol. 10, no. 3, pp. 41–50, Sep. 2015.
- [28] J. An, K. Yang, J. Wu, N. Ye, S. Guo, and Z. Liao, "Achieving sustainable ultra-dense heterogeneous networks for 5G," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 84–90, Dec. 2017.
- [29] K. Yang, N. Yang, N. Ye, M. Jia, Z. Gao, and R. Fan, "Non-orthogonal multiple access: Achieving sustainable future radio access," *IEEE Commun. Mag.*, vol. 57, no. 2, pp. 116–121, Feb. 2019.
- [30] M. Shi, K. Yang, Z. Han, and D. Niyato, "Coverage analysis of integrated sub-6GHz-mmWave cellular networks with hotspots," *IEEE Trans. Commun.*, vol. 67, no. 11, pp. 8151–8164, Nov. 2019.
- [31] Z. Han, M. Hong, and D. Wang, *Signal Processing and Networking for Big Data Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2017.
- [32] J. Elias, F. Martignon, S. Paris, and J. Wang, "Efficient orchestration mechanisms for congestion mitigation in NFV: Models and algorithms," *IEEE Trans. Serv. Comput.*, vol. 10, no. 4, pp. 534–546, Jul. 2017.
- [33] H. Zhang, Y. Xiao, L. X. Cai, D. Niyato, L. Song, and Z. Han, "A multi-leader multi-follower stackelberg game for resource management in LTE unlicensed," *IEEE Trans. Wireless Commun.*, vol. 16, no. 1, pp. 348–361, Jan. 2017.
- [34] H. Zhang, W. Ding, J. Song, and Z. Han, "A hierarchical game approach for visible light communication and D2D heterogeneous network," in *Proc. IEEE Global Commun. Conf.*, Dec. 2016, pp. 1–6.

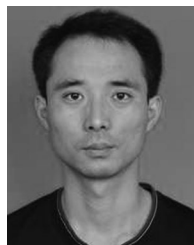
- [35] A. J. Conejo, E. Castillo, R. Minguez, and R. Garcia-Bertrand, *Decomposition Techniques in Mathematical Programming: Engineering and Science Applications*. Berlin, Germany: Springer, 2006.
- [36] J. Cabero, M. J. Ventosa, S. Cerisola, and Á. Baillo, "Modeling risk management in oligopolistic electricity markets: A benders decomposition approach," *IEEE Trans. Power Syst.*, vol. 25, no. 1, pp. 263–271, Feb. 2010.
- [37] L. P. Qian, Y. J. A. Zhang, Y. Wu, and J. Chen, "Joint base station association and power control via benders' decomposition," *IEEE Trans. Wireless Commun.*, vol. 12, no. 4, pp. 1651–1665, Apr. 2013.
- [38] A. Nagarajan and R. Ayyanar, "Design and scheduling of microgrids using benders decomposition," in *Proc. IEEE 43rd Photovolt. Spec. Conf.*, Portland, OR, USA, Jun. 2016, pp. 1843–1847.
- [39] 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.
- [40] L. Liu, X. Chen, M. Bennis, G. Xue, and Z. Han, "A distributed ADMM approach for mobile data offloading in software defined network," in *Proc. IEEE Wireless Commun. Netw. Conf.*, New Orleans, LA, USA, Mar. 2015, pp. 1748–1752.
- [41] L. Liu and Z. Han, "Multi-block ADMM for big data optimization in smart grid," in *Proc. Int. Conf. Comput., Netw. Commun.*, Garden Grove, CA, USA, Feb. 2015, pp. 556–561.
- [42] H. K. Nguyen, Y. Zhang, Z. Chang, and Z. Han, "Parallel and distributed resource allocation with minimum traffic disruption for network virtualization," *IEEE Trans. Commun.*, vol. 65, no. 3, pp. 1162–1175, Mar. 2017.
- [43] H. K. Nguyen, A. Khodaei, and Z. Han, "A big data scale algorithm for optimal scheduling of integrated microgrids," *IEEE Trans. Smart Grid*, vol. 9, no. 1, pp. 274–282, Jan. 2018.
- [44] Y. Chu and Q. Xia, "Generating benders cuts for a general class of integer programming problems," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, J.-C. Régin and M. Rueher, Eds. Berlin, Germany: Springer, 2004, pp. 127–141.
- [45] M. C. P. Paredes, J. J. Escudero-Garzs, and M. J. F.-G. Garca, "PAPR reduction via constellation extension in OFDM systems using generalized benders decomposition and branch-and-bound techniques," *IEEE Trans. Veh. Technol.*, vol. 65, no. 7, pp. 5133–5145, Jul. 2016.
- [46] T. Lin, S. Ma, and S. Zhang, "Iteration complexity analysis of multi-block ADMM for a family of convex minimization without strong convexity," *J. Sci. Comput.*, vol. 69, no. 1, pp. 52–81, 2016.
- [47] N. V. Sahinidis and I. E. Grossmann, "Convergence properties of generalized benders decomposition," *Comput. Chem. Eng.*, vol. 15, no. 7, pp. 481–491, Jul. 1991.
- [48] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Trans. Signal Process.*, vol. 62, no. 7, pp. 1750–1761, Apr. 2014.
- [49] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.
- [50] P. Lubell-Doughtie and J. Sondag, "Practical distributed classification using the alternating direction method of multipliers algorithm," in *Proc. IEEE Int. Conf. Big Data*, Silicon Valley, CA, USA, Oct. 2013, pp. 773–776.
- [51] "Apache Hadoop," Apr. 2015. [Online]. Available: <https://hadoop.apache.org/>
- [52] X. Gao, P. Wang, D. Niyato, K. Yang, and J. An, "Auction-based time scheduling for backscatter-aided RF-powered cognitive radio networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 3, pp. 1684–1697, Mar. 2019.
- [53] T. White, *Hadoop: The Definitive Guide*, 1st ed. Sebastopol, CA, USA: O'Reilly Media, Inc., 2009.
- [54] V. Eramo and F. G. Lavacca, "Computing and bandwidth resource allocation in multi-provider NFV environment," *IEEE Commun. Lett.*, vol. 22, no. 10, pp. 2060–2063, Aug. 2018.



Ye Yu (S'18) received the B.Sc. degree in information engineering, in 2014, from the Beijing Institute of Technology, Beijing, China, where he is currently working toward the Ph.D. degree with the School of Information and Electronics. From September 2016 to September 2018, he was a Visiting Student with the Computer Science Department, University of Houston, Houston, TX, USA. His research interests include green communications, network virtualization, heterogeneous networks, and fog computing.



Xiangyuan Bu received the B.E. and Ph.D. degrees in communications engineering from the Beijing Institute of Technology (BIT), Beijing, China, in 1987 and 2007, respectively. He is currently a Professor with the School of Information and Electronics, BIT. His current research interests include digital signal processing, channel coding theory, MIMO system, space time signal processing, and satellite communications.



Kai Yang (M'12) received the B.E. degree from the National University of Defense Technology, Changsha, China, in 2005 and the Ph.D. degree from the Beijing Institute of Technology, Beijing, China, in 2010, both in communications engineering. From January 2010 to July 2010, he was with the Department of Electronic and Information Engineering, Hong Kong Polytechnic University. From 2010 to 2013, he was with the Alcatel-Lucent Shanghai Bell, Shanghai, China. In 2013, he joined the Laboratoire de Recherche en Informatique, University Paris Sud 11, Orsay, France. He is currently with the School of Information and Electronics, Beijing Institute of Technology. His current research interests include convex optimization, massive MIMO, mmWave systems, resource allocation, and interference mitigation.



Hung Khanh Nguyen received the B.S. degree from the Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam, in 2010, the M.S. degree from Kyung Hee University, Seoul, South Korea, in 2012, and the Ph.D. degree in electrical and computer engineering from the University of Houston, Houston, TX, USA, in 2017. His research interests include big data analytic, resource allocation and game theory, distributed and parallel optimization, large-scale data processing in smart grid and wireless network.



Zhu Han (S'01-M'04-SM'09-F'14) received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 1997, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, MD, USA, in 1999 and 2003, respectively.

From 2000 to 2002, he was an R&D Engineer of JDSU, Germantown, MD, USA. From 2003 to 2006, he was a Research Associate with the University of Maryland. From 2006 to 2008, he was an Assistant Professor with Boise State University, Boise, ID, USA. He is currently a John and Rebecca Moores Professor with the Electrical and Computer Engineering Department as well as with the Computer Science Department, University of Houston, Houston, TX, USA. He is also a Chair Professor with the National Chiao Tung University, Hsinchu, Taiwan, R.O.C. His research interests include wireless resource allocation and management, wireless communications and networking, game theory, big data analysis, security, and smart grid. Dr. Han was the recipient of an NSF Career Award in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society in 2011, the EURASIP Best Paper Award for the *Journal on Advances in Signal Processing* in 2015, IEEE Leonard G. Abraham Prize in the field of communications systems (Best Paper Award in IEEE JSAC) in 2016, and several best paper awards in IEEE conferences. He was an IEEE Communications Society Distinguished Lecturer from 2015 to 2018, and has been an AAAS Fellow since 2019 and ACM Distinguished Member since 2019. He has been 1% highly cited Researcher since 2017 according to Web of Science.