Virtual Network Function Placement Considering Resource Optimization and SFC Requests in Cloud Datacenter

Defang Li, *Student Member, IEEE*, Peilin Hong, Kaiping Xue, *Senior Member, IEEE*, and Jianing Pei

Abstract—Network function virtualization (NFV) brings great conveniences and benefits for the enterprises to outsource their network functions to the cloud datacenter. In this paper, we address the virtual network function (VNF) placement problem in cloud datacenter considering users' service function chain requests (SFCRs). To optimize the resource utilization, we take two less-considered factors into consideration, which are the time-varying workloads, and the basic resource consumptions (BRCs) when instantiating VNFs in physical machines (PMs). Then the VNF placement problem is formulated as an integer linear programming (ILP) model with the aim of minimizing the number of used PMs. Afterwards, a Two-StAge heurisTic solution (T-SAT) is designed to solve the ILP. T-SAT consists of a correlation-based greedy algorithm for SFCR mapping (first stage) and a further adjustment algorithm for virtual network function requests (VNFRs) in each SFCR (second stage). Finally, we evaluate T-SAT with the artificial data we compose with Gaussian function and trace data derived from Google's datacenters. The simulation results demonstrate that the number of used PMs derived by T-SAT is near to the optimal results and much smaller than the benchmarks. Besides, it improves the network resource utilization significantly.

Index Terms—Virtual network function placement, resource optimization, service function chain, time-varying workloads, multi-tenancy, basic resource consumptions, correlation-based algorithm

1 Introduction

Network function virtualization (NFV) [1] allows network functions (NFs) or middleboxes traditionally attached on specific hardwares to be realized in software and dynamically outsourced to be run on any common off-the-shelf servers, which brings great conveniences and flexibilities in programmability, management and policy interactions between NFs. Nowadays, more and more network operators and enterprises leverage NFV technology to reduce the cost of infrastructure construction and management [2], [3].

When hundreds of users outsource their NFs to the cloud [4], [5], virtual network functions (VNFs) placement (also mentioned as VNF instantiation) becomes an important but difficult problem for the cloud service providers (CSPs), which has received more and more attention from academia and industry. In the problem, one or more positions where to instantiate a series of VNFs should be decided with an optimization objective, such as minimizing the resource consumptions [6], [7] and maximizing the number of users' requests that can be served [8]. A good placement solution can improve the utilization efficiency of network resources and reduce the CAPital EXpenditures and OPerating EXpenses (CAPEX/OPEX) greatly, which will produce more profits for the CSPs. Basically, VNF placement problem is usually formulated as an integer programming or linear programming model, and lots of heuristic schemes have been proposed [6], [9], [10], [11].

Generally, a user's service request is accomplished by a service function chain (SFC), which is constructed by a list of VNFs within a specified order [12], [13]. Thus, we treat a user's NF outsourcing request as an SFC request, simplified as SFCR. Meanwhile, to make a distinction, we call the elements in a specific SFCR as VNF requests (VNFRs), corresponding to the VNFs in an SFC. Given SFCRs proposed by different users, the VNF placement problem is specific to how to deploy a series of SFCs, separately containing multiple VNFs, onto physical machines (PMs) in the cloud.

Different from most works in existing, the VNFs are multi-tenant capable in our considerations, which means that one VNF can provide services for multiple tenants at the same time [14]. Specifically, the VNFRs in SFCRs can be seen as the tenants in our paper. To optimize the resource utilization while placing the VNFs, we take the following two factors into consideration:

1) *Consolidations of Time-Varying Workloads*. Generally speaking, the workloads of the NFs are time-varying, and the peaks and valleys in one NF's workload do not necessarily coincide with the others statistically [15], [16]. For a multi-tenant capable VNF, the capacity of the VNF is commensurate with the workloads of all VNFRs on it, which implies that resources are allocated to the VNFs based on workloads of all

• The authors are with the Key Laboratory of Wireless-Optical Communications, Chinese Academy of Sciences, School of Information Science and Technology, University of Science and Technology of China. No. 96 Jinzhai Road, Hefei, Anhui Province 230026, P. R. China.
E-mail: {ldf911, jianingp}@mail.ustc.edu.cn, {plhong, kpxue}@ustc.edu.cn.

Manuscript received 14 Jan. 2017; revised 13 Dec. 2017; accepted 29 Jan. 2018. Date of publication 5 Feb. 2018; date of current version 13 June 2018.
(Corresponding author: Defang Li.)
Recommended for acceptance by B. Ravindran.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2018.2802518

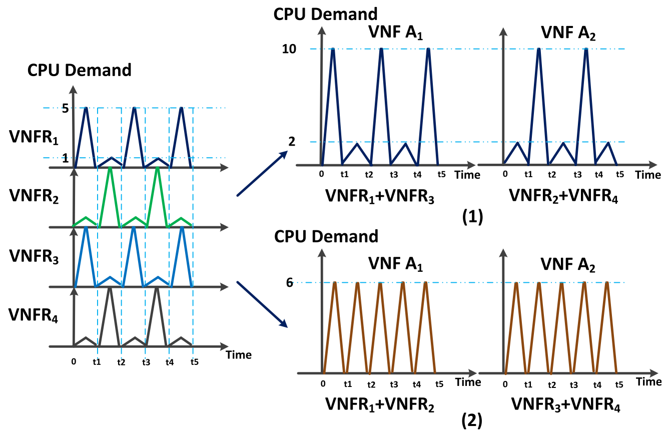

Fig. 1. A schematic diagram of VNFR composition with time-varying workload.

VNFRs on them. So based on the predicted workloads of VNFRs, we can gather together the VNFRs, demanding the same type of VNF and with complementary workload patterns, on the same multi-tenant capable VNF to improve the utilization efficiency of network resources. Referring to the real cases in [15], [16], [17], we use a simple schematic diagram to demonstrate the advantages. As is shown in Fig. 1, the left part shows the time-varying CPU demand of 4 VNFRs for VNF A, the right part shows two combinations of the 4 VNFRs: in combination *1*, VNFR_1 and VNFR_3 are hosted on VNF A_1, VNFR_2 and VNFR_4 are hosted on VNF A_2; in combination *2*, VNFR_1 and VNFR_2 are hosted on VNF A_1, VNFR_3 and VNFR_4 are hosted on VNF A_2. Compared with combination *1*, the workload of combination *2* is more smooth, and the peak value of combination *2* is 6 compared with 10 in combination *1*, so we just need to allocate at most 6 units CPU resources to VNF A_1 and A_2 respectively in combination *2*, compared with 10 units in combination *1*. Thus, we can say that combination *2* leads to a more efficient utilization of PM's resources than combination *1*.

2) *Basic Resource Consumptions and the Sharing*. We consider the basic resource consumptions (BRCs) when instantiating VNFs in a PM. BRCs are the resource consumptions for the basic function maintaining of one software, like the resource consumptions to maintain the OS and related libraries of one VNF [18], [19] before it starts to provide service. Specifically, for the multi-tenant capable VNFs, BRCs are the resource consumptions to maintain the shared application platform and database by the VNFRs on them. We concretize them as the CPU and memory consumptions in this paper, which are the CPU BRCs (c-BRCs) and memory BRCs (m-BRCs) respectively. Also, we assume that BRCs of one VNF are fixed, and they have no relationship with VNF's processing capacity. For a series of VNFRs that are hosted on the same multi-tenant capable VNF, they share the same block of BRCs. By gathering the VNFRs that demand the same type of VNF together, we can save more VNF instances, then BRCs decrease subsequently, leading to fewer node resource consumptions.

As stated before, users' NF requests are accomplished by SFCs, so the flow of one user's request must go through a series of VNFs in a specified order to get served. If all the VNFs that an SFCR needs are in the same PM (Scenario 1), the traffic loads between the VNFs will be restricted in the PM, and the corresponding bandwidth consumptions on the links will not occur. However, if the VNFs needed are distributed among multiple PMs (Scenario 2), the corresponding bandwidth consumptions on the links cannot be avoided. Compared with Scenario 2, Scenario 1 needs more VNF instances, so BRCs are more in Scenario 1. Thus, we can find that there is a confliction between bandwidth consumptions and BRCs, and we need to make a balance between them.

In summary, we consider the VNF placement problem in cloud datacenter with a series of known SFCRs. To optimize the utilization of network resources, we take the time-varying workloads and BRCs into consideration. Afterwards, the VNF placement problem is formulated as an integer linear programming (ILP) model, aiming to minimize the number of used PMs. The VNF placement problem has turned out to be NP-hard [6], [9], [10], [20], [21], so is the VNF placement problem in our paper. Thus we solve the ILP with a small number of SFCRs using Gurobi [22] and propose a Two-StAge heurisTic solution (T-SAT) to solve the ILP when the number of SFCRs is large. T-SAT can be divided into two parts: one is the correlation-based greedy algorithm, which aims to map the SFCRs to the PMs according to the correlation between their predicted workloads, and the other is a further adjustment process based on VNFRs, which aims to gather the VNFRs that demand the same type of VNF together to save more node resources, owing to BRCs sharing among VNFRs on the same multi-tenant capable VNF.

Our major contributions can be summarized as:

1) To improve the utilization efficiency of network resources, we consider the time-varying feature of NF requests' workloads. Specifically, we design a correlation-based algorithm to handle them.
2) We consider BRCs and multi-tenancy of VNFs and reveal the confliction between bandwidth consumptions and BRCs. Then we exploit the fact that VNFRs on the multi-tenant capable VNFs can share BRCs to save more node resources while keeping a balance between bandwidth consumptions and BRCs.
3) We formulate the VNF placement problem as an ILP model and propose a novel two-stage solution, T-SAT. Then T-SAT is evaluated in detail with the artificial data we compose with Gaussian function and the trace data derived from Google's datacenters [23]. The simulation results demonstrate that T-SAT can improve the utilization of network resources significantly and acquire a smaller number of used PMs than the benchmarks.

The remainder of this paper is organized as follows. Section 2 reviews the related works. In Section 3, we describe the system model and the related concepts. Section 4 formulates the VNF placement problem and our solution T-SAT is introduced in Section 5. Afterwards, Section 6 is the performance evaluation of T-SAT. Finally, we conclude the paper in Section 7.

2 Related Work

NFV and the related VNF placement problem have attracted plenty of attention of researchers. In [6], the structures of SFCs are not known forward, and the authors proposed a model for formalizing the SFCs, then they formulated the VNF mapping problem as a Mixed Integer Quadratically Constraints Program (MIQCP). A closed-loop with critical mapping feedback algorithm to jointly optimize the topology design and mapping of multiple SFCs is presented in [7]. In [8], they thought that the relationship between link and server usage plays a crucial role in the joint VNF placement and path selection problem. Then the authors gave a mathematical program analysis about it, and proposed a heuristic solution based on that. In [9], the authors studied the VNFs placement in the datacenter where optical technologies are employed, with the goal of minimizing the expensive optical/electronic/optical (O/E/O) conversion times for NFV chaining. Cohen et al. [11] considered the actual placement of VNFs within the physical network, and provided a near optimal solution with theoretically proven performance. In [24], Rankothge et al. optimized the resource allocation for VNFs in cloud datacenter using genetic algorithms. However, few of the existing works about VNF placement problem pay attention to the multi-tenant capable VNFs, despite the fact that the multi-tenant capable VNFs can lead to a more efficient utilization of network resources.

It is worth noting that the authors in [10] mentioned that more VNFs mean more consumption of the computing resources, but fewer VNFs result in more traffic loads. They pointed out the confliction between the node resource consumptions and bandwidth consumptions, but they did not consider the overhead when instantiating VNFs. In [25], when calculating the resource consumptions of one VM, the authors considered the overhead of virtualization besides the loads of tenants, which has some similarity with the BRCs we consider. However, they did not pay attention to the confliction between the node resource consumptions and bandwidth consumptions.

Although few works consider the time-varying workload in VNF placement problem, we find that plenty of works have been done on the VM consolidation problem considering time-varying workload.

In [16], the authors thought that the peaks and valleys in one workload pattern do not necessarily coincide with the others. Thus, the unused resources of a low utilized VM can be borrowed by the other co-located VMs with high utilization. They proposed a VM selection algorithm that seeks to find those VM combinations with complementary workload patterns based on Pearson's correlation coefficient. Zhang et al. [26] proposed to reserve some extra resources on each PM to accommodate bursty workload. And they used the two-state Markov chain to capture the burstiness of workload and developed a novel server consolidation algorithm. Zheng et al. [27] proposed PowerNets to save the energy consumption in datacenter based on the time-varying workloads. We have the similar view with the authors in [17] that multiplexing VNFs with dynamic workload properly will increase the utilization efficiency of network resources. Besides, we have to consider the interactions between the VNFs at the same time owing to the property of SFCs, which results in the confliction between node resource consumptions and bandwidth consumptions.

Referring to previous works, we formulate the VNF placement problem as an ILP model with the aim of minimizing the number of used PMs while considering the time-varying workloads and BRCs. Then we propose a two-stage heuristic solution to solve it.

3 System Model and Related Concepts

In this section, we introduce the system model and make a further explanation for the concepts referred to before.

3.1 System Model

We represent the substrate network as an undirected graph $G=(N^s, E^s)$, where N^s indicates the set of total nodes in substrate network and E^s indicates the link set of the substrate network. Specifically, we use P to indicate the set of total PMs and R to indicate the set of total access switches, which handle the service accessing of SFCRs.

We assume that there are $|\Gamma|$ SFCRs in total, and use a 4-tuple $(\mathrm{ing}_\gamma, \mathrm{eg}_\gamma, \Psi_\gamma, E^\nu_\gamma)$ to indicate SFCR γ, where ing_γ indicates the ingress node of SFCR γ, eg_γ indicates the egress node of SFCR γ, Ψ_γ indicates the set of total VNFRs in SFCR γ and E^ν_γ indicates the set of logical links between the nodes of SFCR γ. Generally, the ingress node and egress node of the same SFCR should be hosted on the same access switch.

3.2 Multi-Tenancy and BRCs

Multi-tenancy is a software architecture principle in the realm of the Software as a Service (SaaS) business model [14], which allows multiple tenants to share the same software instance [28]. Compared with the single-tenant architecture, in which each tenant gets its own instance of application, multi-tenancy can lead to higher resource utilization, lower service price, and more efficient management for the CSPs. So we assume that the VNFs are multi-tenant capable in this paper. As a result of multi-tenancy, we just need to place one instance of some specified VNF in a PM [14]. But from the view of the whole network, there are multiple instances of one specified VNF.

BRCs are the resource consumptions for the basic function maintaining of one software. Traditionally, a VNF is hosted on one VM, and it needs independent guest OS and libraries [29], so BRCs are the resource consumptions to maintain the functioning of VM platform, guest OS and libraries that one VNF needs. For a multi-tenant capable VNF, the resource consumptions maintaining the shared application platform and database can be seen as BRCs. And the VNFRs on the same multi-tenant capable VNF can share the BRCs, even if they belong to different SFCRs. Besides, we assume that BRCs of one VNF are fixed and they have no relationship with the processing capacity of that VNF.

3.3 Correlation Coefficients

We need to evaluate the correlation between the workloads of different SFCRs or VNFRs, avoiding that SFCRs or VNFRs whose workloads have high correlation are placed together. By saying the two workloads have high correlation, we mean that their peaks and valleys are coincide with each other frequently. To evaluate the correlation, we

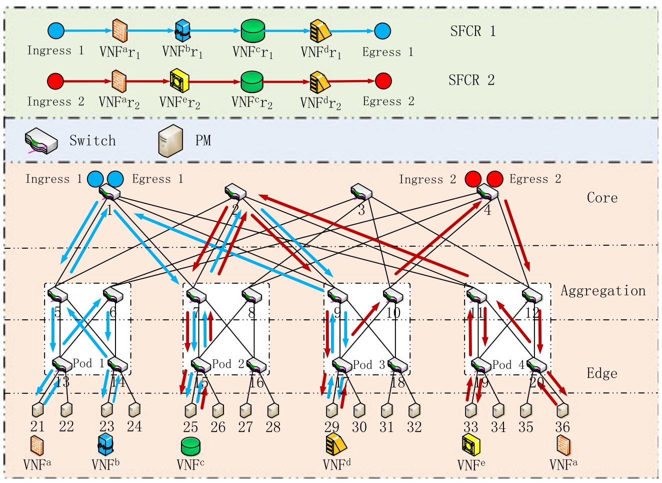

Fig. 2. A VNF placement instance with a 3-layer fat-tree topology.

should compare the workloads within a period. So every workload variable within time interval T is corresponding to a sequence of values. Mathematically speaking, workload variables are vectors instead of constants.

Several coefficients are often used to evaluate the correlation between two vectors, such as Pearson's correlation coefficient [16], [30], Minkowski Distance [31], and Cosine correlation coefficient [17]. We apply the above three mentioned correlation coefficients to T-SAT respectively, and find that the Cosine correlation coefficient can acquire the best effect from plenty of experiments. So we choose it as the final candidate to T-SAT.

The Cosine correlation coefficient is as follows:

$$\cos_{XY} = \frac{\sum_{i=1}^{n} X_i Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2}\sqrt{\sum_{i=1}^{n} Y_i^2}}. \tag{1}$$

The bigger the $\cos_{XY}$ is, the more related vectors X and Y are. In our paper, we need to gather the SFCRs or VNFRs with complementary workload patterns together. Hence the smaller the $\cos_{XY}$ is, the more likely the two SFCRs or VNFRs are mapped in the same PM.

4 Problem Statement

In this section, we give a description about the VNF placement problem we consider through a simple case, and complete the formulations about the problem.

4.1 Problem Description

Fig. 2 shows a non-trivial example about how to place VNFs for two SFCRs in a 3-layer fat-tree topology. For the fat-tree topology, the switches in the core layer (core switches) are responsible for the service accessing. In the figure, VNF^{i}r indicates different VNFR, corresponding to VNF^i that indicates different types of VNFs respectively, $i \in \{a, b, c, d, e\}$; SFCR 1 enters the datacenter from switch 1, and SFCR 2 from switch 4. We assume that VNF^{a}r$_1$ is mapped on PM 21; VNF^{b}r$_1$ on PM 23; VNF^{c}r$_1$ on PM 25; VNF^{d}r$_1$ on PM 29; VNF^{a}r$_2$ on PM 36; VNF^{e}r$_2$ on PM 33; VNF^{c}r$_2$ on PM 25; VNF^{d}r$_2$ on PM 29; Then we need to place two instances of VNF^a on PM 21 and 36, respectively; an instance of VNF^b on PM 23; an instance of VNF^c on PM 25; an instance of VNF^d on PM 29; an instance of VNF^e on PM 33. And the total BRCs are 6 shares. The flow path of SFCR 1 is $1 \rightarrow 5 \rightarrow 13 \rightarrow 21 \rightarrow 13 \rightarrow 6 \rightarrow 14 \rightarrow 23 \rightarrow 14 \rightarrow 5 \rightarrow 1 \rightarrow 7 \rightarrow 15 \rightarrow 25 \rightarrow 15 \rightarrow 7 \rightarrow 2 \rightarrow 9 \rightarrow 17 \rightarrow 29 \rightarrow 17 \rightarrow 9 \rightarrow 1$, which is demonstrated in blue arrows. The flow path of SFCR 2 is $4 \rightarrow 12 \rightarrow 20 \rightarrow 36 \rightarrow 20 \rightarrow 11 \rightarrow 19 \rightarrow 33 \rightarrow 19 \rightarrow 11 \rightarrow 2 \rightarrow 7 \rightarrow 15 \rightarrow 25 \rightarrow 15 \rightarrow 7 \rightarrow 2 \rightarrow 9 \rightarrow 17 \rightarrow 29 \rightarrow 17 \rightarrow 10 \rightarrow 4$, which is demonstrated in red arrows. If we change the mapping position of VNF^{a}r$_2$ from PM 36 to PM 21, we do not need to place the instance of VNF^a on PM 36, so PM 36 is free out and the total BRCs become 5 shares. However, the flow path of SFCR 2 becomes $4 \rightarrow 6 \rightarrow 13 \rightarrow 21 \rightarrow 13 \rightarrow 5 \rightarrow 1 \rightarrow 11 \rightarrow 19 \rightarrow 33 \rightarrow 19 \rightarrow 11 \rightarrow 2 \rightarrow 7 \rightarrow 15 \rightarrow 25 \rightarrow 15 \rightarrow 7 \rightarrow 2 \rightarrow 9 \rightarrow 17 \rightarrow 29 \rightarrow 17 \rightarrow 10 \rightarrow 4$. The number of links that SFCR 2 passes through increases from 22 to 24, so the bandwidth consumptions increase. If we change the mapping position of VNF^{c}r$_2$ from PM 25 to PM 29, we need to instantiate another instance of VNF^c on PM 29 and the BRCs become 7 shares. However, the flow path of SFCR 2 becomes $4 \rightarrow 12 \rightarrow 20 \rightarrow 36 \rightarrow 20 \rightarrow 11 \rightarrow 19 \rightarrow 33 \rightarrow 19 \rightarrow 12 \rightarrow 3 \rightarrow 10 \rightarrow 17 \rightarrow 29 \rightarrow 17 \rightarrow 10 \rightarrow 4$. The number of links that SFCR 2 passes through decreases from 22 to 16, so the bandwidth consumptions decrease.

The above simple case shows a VNF placement process and reveals the confliction between BRCs and bandwidth consumptions. Generally speaking, we accomplish the mapping of the SFCRs or VNFRs first, and then based on the mapping results, we determine the placement of the related VNFs and allocate resources to them commensurate with the workloads of VNFRs on them. It is worth noting that the serving capacity of one VNF is determined by the VNFRs it serves, which is a constant value once the placement of VNFs is accomplished. Moreover, as stated before, the time-varying workloads also have to be considered to improve the utilization efficiency of the network resources.

The activating of one PM is usually associated with much energy and capital cost, and fewer PMs mean fewer costs and motivate the increasing of the network resource utilization. So in this paper, our target is to minimize the number of used PMs. Considering the constraints of the limited resources, namely CPU, memory and bandwidth, we formulate the VNF placement problem as an integer linear programming model.

4.2 Problem Formulation

The notations used in the following equations are described in Table 1.

First, we must ensure that one node in an SFCR is mapped on one and only one node in the substrate network, so for SFCR γ

$$\sum_{u=0}^{|P|-1} x_{\gamma, n_i^v, n_u^s} = 1, n_u^s \in P, n_i^v \in \Psi_\gamma \setminus \{\mathrm{ing}_\gamma, \mathrm{eg}_\gamma\}, \tag{2a}$$

$$\sum_{u=0}^{|R|-1} x_{\gamma, n_i^v, n_u^s} = 1, n_u^s \in R, n_i^v \in \{\mathrm{ing}_\gamma, \mathrm{eg}_\gamma\}, \tag{2b}$$

$$x_{\gamma, \mathrm{ing}_\gamma, n_u^s} = x_{\gamma, \mathrm{eg}_\gamma, n_u^s}, n_u^s \in R, \tag{2c}$$

where $\gamma \in \Gamma$. Eq. (2a) restricts that one VNFR can only be mapped on one PM. Eqs. (2b) and (2c) restrict that ing_γ and

TABLE 1
Notations

Pramaters	Descriptions
SFCR related	
Γ	set of total SFCRs, $\gamma \in \Gamma$ is an SFCR.
ing_γ	ingress node of SFCR γ.
eg_γ	egress node of SFCR γ.
Ψ_γ	set of total VNFRs in SFCR γ.
E^v_γ	set of logical links between the nodes of SFCR γ.
n^v_i, n^v_j	two nodes in one SFCR.
(n^v_i, n^v_j)	logical link between n^v_i and n^v_j, $(n^v_i, n^v_j) \in E^v_\gamma$.
Φ	types number of VNFs, $\phi \in \{0, 1, \dots, \Phi - 1\}$ is an element.
Topology related	
N^s	set of total nodes in substrate network.
P	set of total PMs.
R	set of total access switches.
E^s	link set of the substrate network.
n^s_u, n^s_v	two nodes in the substrate network.
(n^s_u, n^s_v)	substrate link between n^s_u and n^s_v.
Sampling related	
T	time window length of the workload we consider.
t_Δ	sample interval of the workload.
Λ	sample times to the workload in time window T. λ indicates the sequence number of the sampling, $\lambda \in \{0, 1, \dots, \Lambda - 1\}$.
Resource related	
$\text{cpu}_{\gamma,n^v_i}(\lambda t_\Delta)$	CPU consumption by VNFR n^v_i in SFCR γ at time λt_Δ.
$\text{mem}_{\gamma,n^v_i}(\lambda t_\Delta)$	memory consumption by VNFR n^v_i in SFCR γ at time λt_Δ.
$b_{\gamma,n^v_i,n^v_j}(\lambda t_\Delta)$	bandwidth consumption by logical link $(n^v_i\ n^v_j)$ in SFCR γ at time λt_Δ.
$\text{brc}^{\text{cpu}}_\phi$	BRCs of CPU when placing an instance of VNF ϕ, $\phi \in \Phi$.
$\text{brc}^{\text{mem}}_\phi$	BRCs of memory when placing an instance of VNF ϕ, $\phi \in \Phi$.
$C^{\text{cpu}}_{n^s_u}$	CPU capacity of PM n^s_u.
$C^{\text{mem}}_{n^s_u}$	memory capacity of PM n^s_u.
$C^{\text{link}}_{(n^s_u,n^s_v)}$	link capacity of substrate link (n^s_u, n^s_v).
$\theta_{\text{cpu}}, \theta_{\text{mem}}, \theta_{\text{link}}$	resource violation threshold for CPU, memory, and link within time interval T, respectively.
Binary variables	
x_{γ,n^v_i,n^s_u}	whether n^v_i in SFCR γ is hosted on substrate node n^s_u, $n^s_u \in P \cup R$.
$l_{\gamma,n^v_i,\phi}$	whether VNFR n^v_i in SFCR γ demands VNF ϕ.
$y_{\gamma,n^v_i,n^v_j,n^s_u,n^s_v}$	whether the logical link (n^v_i, n^v_j) in SFCR γ is mapped on substrate link (n^s_u, n^s_v).
z_{ϕ,n^s_u}	whether an instance of VNF ϕ is placed on PM n^s_u, $n^s_u \in P$.
$h_{n^s_u}$	whether PM n^s_u is activated, $n^s_u \in P$.

eg_γ of SFCR γ can only be mapped on the same access switch.

Then we should ensure that the total resource consumptions by the VNFRs in one PM should not exceed the resource capacity of that PM at each sampling time λt_Δ, where t_Δ is the sample interval of the workload. So the constraint for CPU consumptions is

$$\begin{aligned}\sum_{\gamma=0}^{|\Gamma|-1}\sum_{i=0}^{|\Psi_\gamma|-1} &\text{cpu}_{\gamma,n^v_i}(\lambda t_\Delta)\cdot x_{\gamma,n^v_i,n^s_u}\\ &+\sum_{\phi=0}^{\Phi-1}\text{brc}^{\text{cpu}}_\phi \cdot z_{\phi,n^s_u} \le C^{\text{cpu}}_{n^s_u}, n^s_u \in P,\end{aligned} \tag{3}$$

where z_{ϕ,n^s_u} indicates whether an instance of VNF ϕ is placed on PM n^s_u. In Eq. (3), $\sum_{\phi=0}^{\Phi-1}\text{brc}^{\text{cpu}}_\phi \times z_{\phi,n^s_u}$ indicates the total CPU BRCs in PM n^s_u.

z_{ϕ,n^s_u} relies on x_{γ,n^v_i,n^s_u} and $l_{\gamma,n^v_i,\phi}$

$$z_{\phi,n^s_u} = \begin{cases} 1, & \sum_{\gamma=0}^{|\Gamma|-1}\sum_{i=0}^{|\Psi_\gamma|-1} x_{\gamma,n^v_i,n^s_u}\cdot l_{\gamma,n^v_i,\phi} \ge 1,\\ 0 & \textit{otherwise}, \end{cases} \tag{4}$$

where $n^s_u \in P$, and $l_{\gamma,n^v_i,\phi}$ indicates whether VNFR n^v_i in SFCR γ demands VNF ϕ, if so, $l_{\gamma,n^v_i,\phi} = 1$. So Eq. (4) indicates that we need to place an instance of VNF ϕ on PM n^s_u for the VNFRs that demand it.

Like Eq. (3) for CPU, we can formulate the memory constraints as

$$\begin{aligned}\sum_{\gamma=0}^{|\Gamma|-1}\sum_{i=0}^{|\Psi_\gamma|-1} &\text{mem}_{\gamma,n^v_i}(\lambda t_\Delta)\cdot x_{\gamma,n^v_i,n^s_u}\\ &+\sum_{\phi=0}^{\Phi-1}\text{brc}^{\text{mem}}_\phi \cdot z_{\phi,n^s_u} \le C^{\text{mem}}_{n^s_u},\end{aligned} \tag{5}$$

where $n^s_u \in P$.

In fact, for a time period, the times of resource over-utilization can be tolerated within certain range. Then we define the following binaries:

$$I_{\text{cpu}}(\lambda t_\Delta) = \begin{cases} 1, & \textit{Eq. 3 is violated},\\ 0 & \textit{otherwise}, \end{cases} \tag{6}$$

$$I_{\text{mem}}(\lambda t_\Delta) = \begin{cases} 1, & \textit{Eq. 5 is violated},\\ 0 & \textit{otherwise}, \end{cases} \tag{7}$$

where $I_{\text{cpu}}(\lambda t_\Delta)$ and $I_{\text{mem}}(\lambda t_\Delta)$ indicate whether the CPU and memory constraints are violated at sampling time λt_Δ, respectively. Therefore, the CPU and memory constraints within time interval T can be

$$\sum_{\lambda=0}^{\Lambda-1} I_{cpu}(\lambda t_\Delta)/\Lambda \le \theta_{\text{cpu}}, \tag{8}$$

$$\sum_{\lambda=0}^{\Lambda-1} I_{mem}(\lambda t_\Delta)/\Lambda \le \theta_{\text{mem}}, \tag{9}$$

where θ_{cpu} and θ_{mem} indicate the resource violation threshold for CPU and memory, respectively. Eqs. (8) and (9) indicate that the total resource violation times within time interval T must not exceed a threshold. The relationship between T, t_Δ, and Λ is

$$t_\Delta = T/\Lambda. \tag{10}$$

Following the node constraints, we introduce the link constraints subsequently.

First, the link variable is

$$y_{\gamma,n_i^v,n_j^v,n_u^s,n_v^s} = \begin{cases} 1, & \textit{condition is satisfied}, \\ 0 & \textit{otherwise}, \end{cases} \tag{11}$$

where the *condition* in Eq. (11) is that logical link (n_i^v, n_j^v) in SFCR γ is mapped on substrate link (n_u^s, n_v^s), $(n_u^s, n_v^s) \in E^s$. In general, each logical link is corresponding to a path in the substrate network. However, in the VNF placement problem, two different nodes in one SFCR can be mapped on the same substrate node, and then the corresponding logical link is limited in one PM.

Similar to the constraints about node resources, we have the following link resource constraint at sampling time λt_Δ

$$\sum_{\gamma=0}^{|\Gamma|-1} \sum_{(n_i^v,n_j^v)\in E_\gamma^v} b_{\gamma,n_i^v,n_j^v}(\lambda t_\Delta) \cdot y_{\gamma,n_i^v,n_j^v,n_u^s,n_v^s} \le C^{\text{link}}_{(n_u^s,n_v^s)}, \tag{12}$$

where $(n_u^s, n_v^s) \in E^s$ and $\gamma \in \Gamma$.

The link resource constraint within time interval T is

$$I_{\text{link}}(\lambda t_\Delta) = \begin{cases} 1, & \textit{Eq. 12 is violated}, \\ 0 & \textit{otherwise}, \end{cases} \tag{13}$$

$$\sum_{\lambda=0}^{\Lambda-1} I_{\text{link}}(\lambda t_\Delta)/\Lambda \le \theta_{\text{link}}, \tag{14}$$

where θ_{link} indicates the resource violation threshold for link resources within time interval T.

For the flow of SFCR γ, the following constraints must be satisfied

$$\sum_{n_v^s}^{N^s} y_{\gamma,n_i^v,n_j^v,n_u^s,n_v^s} \le 1, \tag{15a}$$

$$\sum_{n_v^s}^{N^s} y_{\gamma,n_i^v,n_j^v,n_v^s,n_u^s} \le 1, \tag{15b}$$

$$\sum_{n_v^s}^{N^s} y_{\gamma,n_i^v,n_j^v,n_u^s,n_v^s} - \sum_{n_v^s}^{N^s} y_{\gamma,n_i^v,n_j^v,n_v^s,n_u^s} = x_{\gamma,n_i^v,n_u^s} - x_{\gamma,n_j^v,n_u^s}, \tag{15c}$$

where $(n_v^s, n_u^s) \in E^s,\ (n_i^v, n_j^v) \in E_\gamma^v,\ \gamma \in \Gamma$.

Eq. (15a) indicates whether logical link (n_i^v, n_j^v) in SFCR γ is mapped on one of the substrate links that leave out node n_u^s, if so, $\sum_{n_v^s}^{N^s} y_{\gamma,n_i^v,n_j^v,n_u^s,n_v^s}$ equals 1. Eq. (15b) indicates whether the same logical link is mapped on one of the substrate links that go in node n_u^s, if so, $\sum_{n_v^s}^{N^s} y_{\gamma,n_i^v,n_j^v,n_v^s,n_u^s}$ equals 1. The two equations also ensure that one logical link cannot be split. Eq. (15c) ensures that the path in the substrate network is consecutive for a flow.

Finally, we use $h_{n_u^s}$ to indicate whether PM n_u^s is activated, $n_u^s \in P$. $h_{n_u^s}$ relies on x_{γ,n_i^v,n_u^s}

$$h_{n_u^s} = \begin{cases} 1, & \sum_{\gamma=0}^{|\Gamma|-1} \sum_{i=0}^{|\Psi_\gamma|-1} x_{\gamma,n_i^v,n_u^s} \ge 1, n_u^s \in P, \\ 0 & \textit{otherwise}, \end{cases} \tag{16}$$

Eq. (16) indicates that even if only one VNFR is mapped on the PM, the PM has also to be activated. The total number of activated PMs is

$$H_t = \sum_{u=0}^{|P|-1} h_{n_u^s}. \tag{17}$$

Our target is to minimize the number of activated PMs

$$\begin{gathered} \min H_t, \\ \textit{s.t. Eq. 2 to Eq. 17.} \end{gathered} \tag{18}$$

The VNF placement problem is formulated as an integer linear programming model, which is NP-hard. For the ILP, we can derive the optimal results by existing optimization solvers, like Gurobi [22], when the problem scale is small. However, it is infeasible to get the optimal results in foreseeable time when the problem scale is large. So we design a Two-StAge heurisTic solution to solve the ILP in polynomial time.

5 Proposed Algorithm

In this section, we present the two-stage heuristic solution, T-SAT. A general idea about T-SAT is presented first in Section 5.1, then the details about the algorithms are demonstrated in Sections 5.2 and 5.3.

5.1 Framework of T-SAT

T-SAT consists of a correlation-based greedy algorithm for mapping SFCRs to PMs and a further adjustment process to reduce BRCs by gathering the VNFRs that demand the same type of VNF together.

We map all the SFCRs one by one to the PMs using the correlation-based greedy algorithm at the first stage. During the process, one SFCR is mapped on one PM as a whole unless the total resources demanded by the SFCR cannot be satisfied by the PM. For those SFCRs that cannot be mapped onto one PM as a whole, we split each of them into several VNFRs and map these VNFRs one by one to the PMs.

If we place VNFs based on the above results, BRCs will occupy a big share of the resources in the PMs. Because each PM has to instantiate almost all kinds of VNFs in it and different VNF instances cannot share BRCs. On the contrary, the interacting traffic between PMs is the least in above situation.

We can reduce BRCs by reducing the number of reduplicated VNF instances. So we gather the VNFRs demanding the same type of VNF together into a smaller group of PMs in the second stage, based on above SFCRs mapping results. Then we can instantiate fewer VNFs, and BRCs are reduced. Subsequently, the total resource consumptions in PMs are fewer, and the number of used PMs decreases. However, the above processes may increase the interacting traffic between the PMs. So we must ensure that the link capacity constraints are not violated during the adjustment process.

5.2 Correlation-Based Greedy Algorithm

Algorithm 1 is the framework of the correlation-based greedy algorithm. First, we establish the correlation coefficient matrix Υ about all SFCRs. Υ is a $|\Gamma| \times |\Gamma|$ square symmetric matrix. $\mu_{i,j}$ is one of its elements, which indicates the correlation coefficient between the workloads of SFCR i and SFCR j. The workload of one SFCR is the linear accumulation of all VNFRs' workloads in it. We set $\mu_{i,j}$ as an infinity value when $i = j$. Otherwise, $\mu_{i,j}$ is calculated by Eq. (20).

Algorithm 1. Correlation-Based Greedy Algorithm

```
1: Input: The set of all the SFCRs: $\Gamma$;
          The initial status of datacenter: $S_0$;
2: Output: Number of used PMs: $P_0$;
          Status of datacenter: $S_1$;
3: Establish the correlation coefficient matrix $\Upsilon$ about all
   SFCRs.
4: while $\Gamma$ is not empty do
5:     Start a new PM, $P_0 = P_0 + 1$.
6:     Pick out the pair of SFCRs in $\Upsilon$ whose workloads have
       the minimal correlation coefficient.
7:     Select the SFCR that consumes more resources as the
       chosen SFCR.
8:     Make a copy of current $\Gamma$ as $\Gamma_c$.
9:     while $\Gamma_c$ is not empty do
10:        if PM can hold the chosen SFCR then
11:            Put the chosen SFCR into the PM.
12:            Remove the chosen SFCR from $\Gamma$ and $\Gamma_c$.
13:            if $\Gamma_c$ is not empty then
14:                Take the SFCRs in the PM as a whole,
                   indicated as $\xi$.
15:                Calculate the workload correlation
                   coefficients between $\xi$ and each of the
                   rest SFCRs.
16:                Pick out the SFCR whose workload has the
                   minimal correlation with the workload of $\xi$ to
                   be mapped nextly.
17:            else
18:                break.
19:            end
20:        else
21:            Decompose the chosen SFCR into VNFRs
               and try to map them into PMs that
               have VNFRs in themselves based on first-fit.
22:            if the above process is successful then
23:                Remove the chosen SFCR from $\Gamma_c$ and $\Gamma$.
24:            else
25:                Remove the chosen SFCR from $\Gamma_c$.
26:            end
27:            if $\Gamma_c$ is not empty then
28:                Pick out the SFCR that has the minimal
                   workload correlation coefficient with $\xi$
                   to be mapped nextly.
29:            else
30:                break.
31:            end
32:        end
33:    end
34: end
35: return $P_0$, $S_1$.
```

As stated before, we consider the constraints of three types of resources, namely CPU, memory and bandwidth. So there are three vectors for each SFCR mathematically, and the mathematic description for the workload of SFCR i can be formulated as

$$\mathcal{M}_i = \begin{pmatrix} \mathrm{cpu}_i^0 & \mathrm{cpu}_i^1 & \dots & \mathrm{cpu}_i^{\Lambda-1} \\ \mathrm{mem}_i^0 & \mathrm{mem}_i^1 & \dots & \mathrm{mem}_i^{\Lambda-1} \\ \mathrm{band}_i^0 & \mathrm{band}_i^1 & \dots & \mathrm{band}_i^{\Lambda-1} \end{pmatrix}, \tag{19}$$

where Λ is the sample times we apply to the workload.

For Eq. (19), the three rows of $\mathcal{M}_i$ indicate the vector variables of CPU, memory and bandwidth respectively. The correlation coefficient $\mu_{i,j}$ between SFCR i and SFCR j can be calculated as

$$\mu_{i,j} = f(\mathcal{M}_i^0, \mathcal{M}_j^0) + f(\mathcal{M}_i^1, \mathcal{M}_j^1) + f(\mathcal{M}_i^2, \mathcal{M}_j^2), \tag{20}$$

where f is the function referred to in Section 3.3 to calculate the correlation coefficient of two vectors.

After establishing the correlation coefficient matrix, we pick out the pair of SFCRs, the workloads of which have the minimal correlation coefficient. Then the SFCR in the pair that consumes more resources is selected and mapped in the PM. At line 8 in Algorithm 1, we make a copy of current SFCR set Γ, indicated as Γ_c, aiming to traverse all the rest SFCRs and check out whether the chosen SFCR can be mapped in the current PM (lines 9-33 in Algorithm 1).

If the chosen SFCR can be hosted in current PM as a whole, we put it in the PM and remove it from both Γ and Γ_c (lines 11-12 in Algorithm 1). After that, we choose the next SFCR to be mapped (lines 14-16 in Algorithm 1). First, the SFCRs already in the PM are taken as a whole, indicated as ξ. Then we calculate the correlation coefficient between the workload of ξ and the workload of each SFCR in Γ_c. At last, the SFCR in Γ_c whose workload has the minimal correlation with the workload of ξ is chosen to be mapped nextly.

If the chosen SFC cannot be hosted in current PM as a whole, we decompose the SFCR into several individual VNFRs and try to map these VNFRs in PMs that already have VNFRs in based on first-fit algorithm (lines 21-26 in Algorithm 1). If these VNFRs cannot be mapped totally, it indicates that the chosen SFCR needs an empty PM to be mapped. So we keep it in Γ to be mapped in future and only remove it from Γ_c. Then if Γ_c is not empty, the SFCR in Γ_c whose workload has the minimal correlation with the workload of ξ is chosen to be mapped nextly.

The above processes are repeated until Γ_c is empty. After the breaking of the inner *while* loop (lines 9-33 in Algorithm 1), we need to check if Γ is empty. If Γ is empty, it means that all the SFCRs are mapped on the PMs.

Based on the above SFCR mapping results, each used PM contains almost all kinds of VNFRs. If we place the related VNF instances based on that, each of PMs has to instantiate almost all types of VNFs, which will results in a great volume of BRCs. However, there is little interacting traffic between the PMs in this situation, so the utilization of bandwidth resource is low relatively. Then the next step is to reduce BRCs with the increasing of the bandwidth utilization, trying to reduce the number of used PMs further.

5.3 Two-Stage Adjustment Algorithm

In the second stage, we design an adjustment algorithm to reduce BRCs. The adjustment algorithm is divided into two parts. The first step tries to gather the VNFRs demanding the same type of VNF together into a smaller number of PMs intra the same cluster, where a cluster is a collection of PMs that are close to each other. By gathering the VNFRs demanding the same type of VNF together, we can reduce the number of reduplicated VNF instances, and then reduce the volume of BRCs. After the adjustment in each cluster, we sort all the PMs in ascending order based on the average resource utilization of each PM, and design an algorithm moving out the VNFRs from the PMs of low resource utilization to the PMs of high resource utilization. If all the VNFRs are moved out from one PM, the PM is released.

5.3.1 *Intra Cluster Adjustment*

The first step is to gather together the VNFRs demanding the same type VNF intra each cluster. To state our algorithm more briefly, we treat all those VNFRs in the same PM that demand the same type of VNF as a whole, indicated as VNFR$_w$. When we migrate a VNFR$_w$ from one PM to another, we mean that all the corresponding VNFRs demanding the same type VNF are migrated.

Algorithm 2 gives the adjustment process intra each of the clusters. First, we calculate the traffic going through each VNFR$_w$ (line 3 in Algorithm 2), which is indicated as VNFR^t_w. VNFR^t_w is the accumulation of the traffic flowing in and out of all VNFRs belonging to the corresponding VNFR$_w$. After the calculation of all the VNFR^t_ws, we pick out the VNFR$_w$ that has the minimal VNFR^t_w in each of the PMs (line 10 in Algorithm 2) and store them in a list, which is indicated as Δ. Nextly, the target PMs are chosen for each of these VNFR$_w$s (line 11 in Algorithm 2), and these VNFR$_w$s are moved to the target PMs, respectively.

Algorithm 2. Intra Cluster Adjustment Algorithm

```
1: Input: Number of used PMs: $P_0$,
          Status of datacenter: $S_1$;
2: Output: Number of used PMs: $P_1$,
           Status of datacenter: $S_2$;
3: Calculate the traffic going through each VNFR$_w$ in each PM,
   indicated as VNFR$^t_w$.
4: Calculate the number of used clusters $N_c$.
5: $i = 0$.
6: while $i < N_c$ do
7:     Store the PMs in cluster $\zeta_i$ to a list $L_{\mathrm{PM}}$.
8:     Backup the status of datacenter.
9:     while $L_{\mathrm{PM}}$ is not empty do
10:        Find the VNFR$_w$ that has the minimal VNFR$^t_w$ in each
           of the PMs belonging to cluster $\zeta_i$, indicated as
           $\Delta = \{\Delta_0, \Delta_1 \ldots \Delta_{\eta-1}\}$, $\eta$ is the size of the cluster.
11:        Choose the target PM for each VNFR$_w$ in $\Delta$. $\leftarrow$
           Function 1
12:        Migrate each VNFR$_w$ in $\Delta$ to the target PM.
13:        if eliminate_exceed() is true then
14:            Check all the PMs in the cluster $\zeta_i$, remove the
               unqualified PM from $L_{\mathrm{PM}}$.
15:        else
16:            Restore the status of datacenter.
17:            Break.
18:        end
19:    end
20:    $i = i + 1$.
21: end
22: return $P_1$, $S_2$.
```

The details about how to find the target PM for each VNFR$_w$ in Δ are shown in Function 1. For a VNFR$_w$ in Δ, which is indicated as Δ_j, the target PM must have the VNFR$_w$ that demands the same type of VNF with Δ_j. Then we take two factors into consideration, which are the VNFR^t_w of the VNFR$_w$ in the target PM (line 9 in Function 1) and the correlation between the workload of Δ_j and the workload of the total VNFRs already in the target PM (line 7 and 13-14 in Function 1). We quantify the above two considerations as a series of weighted factors, ω (line 18 in Function 1), and ω is the weighted difference between normalized VNFR^t_w and normalized correlation coefficient. Finally, the PM with the biggest ω is chosen as the target PM for Δ_j. In this way, on the one hand, we avoid the moving of the VNFR$_w$ with larger VNFR^t_w, incurring as less interacting traffic between PMs as possible. On the other hand, it leads to higher utilization of the resources in PMs, owing to the less correlation between the workload of Δ_j and the workload of the total VNFRs already in the target PM. α and β are the adjustment factors in the calculation of ω.

Function 1. Choose the Target PM

```
1: Input: Status of datacenter: $S_2$, $\Delta$;
2: Output: The target PMs list: $L_t$;
3: $j = 0$.
4: while $j < \eta$ do
5:     for each PM in cluster $\zeta_i$ do
6:         if $\Delta_j$ in the PM then
7:             Calculate the resource demand matrix $\mathcal{M}_j$ for $\Delta_j$.
8:         else if the PM has the same kind VNFR$_w$ with $\Delta_j$ then
9:             Record VNFR$^t_w$ of VNFR$_w$ in the PM as $t_\epsilon$, and
               store it in $T_\epsilon$.
10:
11:    end
12:    for each PM in cluster $\zeta_i$ that has the same kind VNFR$_w$ with
       $\Delta_j$ do
13:        Establish the resource demand matrix for the VNFRs
           already in the PM, indicated as $\mathcal{M}_u$.
14:        Calculate the correlation coefficient between $\mathcal{M}_j$ and
           $\mathcal{M}_u$ as $\mu_\epsilon$, then store it in $\Upsilon_\epsilon$.
15:    end
16:    Normalize all the values in $T_\epsilon$ and $\Upsilon_\epsilon$.
17:    for each pair of values for one PM in $T_\epsilon$ and $\Upsilon_\epsilon$ do
18:        Calculate the weighted factor $\omega$,
                    $\omega = \alpha \times t_\epsilon - \beta \times \mu_\epsilon$.
19:    end
20:    Pick out the PM that has the biggest $\omega$ into $L_t$.
21:    $j = j + 1$.
22: end
23: return $L_t$.
```

By gathering the VNFR$_w$s demanding the same type of VNF together, some PMs do not need to place the corresponding VNF instances. Therefore BRCs in these PMs are saved. However, the above processes may cause resource over-utilization on some PMs. So function **eliminate_exceed()** (line 13 in Algorithm 2) is designed to eliminate the resource over-utilization. The main idea of **eliminate_exceed()** is to migrate part of the VNFRs from the PM with resource over-utilization to other PMs that have free resources. Only if there is no resource over-utilization in all PMs belonging to the current cluster, **eliminate_exceed()** returns *True*. If not, we abandon all the above processes, and restore the status of the datacenter (line 16 in Algorithm 2), then start to deal with next cluster (lines 17 and 20 in Algorithm 2).

For line 14 in Algorithm 2, the PM is unqualified if one of the following conditions is satisfied:

(1) *Condition 1*: The average residual bandwidth of the link that connects the PM directly is less than half of the original value, where the original value is the residual bandwidth after the first stage.

(2) *Condition 2*: All the VNFRs in the PM are moved out.

Condition 1 is to ensure that there is still bandwidth left to execute the second adjustment process. *Condition 2* is to free out the empty PM from the adjustment process.

After implementing Algorithm 2, almost each PM has some free CPU and memory resources. The next step is to gather these free resources together to empty more PMs.

5.3.2 Inter Clusters Adjustment

Algorithm 3 along with Procedures 1 and 2 describes the second adjustment process in detail. First the used PMs are sorted based on their average resource utilization in ascending order, then we try to move all VNFRs one by one from the PMs of low resource utilization to the PMs of high resource utilization.

Algorithm 3. Inter Clusters Adjustment Algorithm

```
1:  Input: Number of used PMs: $P_1$,
           Status of datacenter: $S_2$;
2:  Output: Number of used PMs: $P_2$,
            Status of datacenter: $S_3$;
3:  Calculate the average resource utilization of all used PMs.
4:  Put all used PMs into a list $P_l$, and sort them in ascending
    order based on the average resource utilization.
5:  Set two pointers, $\rho_b$ and $\rho_e$. $\rho_b$ points to the source PM, indi-
    cated as PM$_s$, at the beginning of $P_l$. $\rho_e$ points to the desti-
    nation PM, indicated as PM$_d$, at the end of $P_l$, and a shift
    value $\delta$, $\delta = 0$ initially.
6:  while 1 do
7:      Backup the current status of datacenter.
8:      while $\rho_b < \rho_e$ do
9:          for each VNFR in PM$_s$ do
10:             if PM$_d$ can hold the VNFR then
11:                 Migrate the VNFR to PM$_d$.
12:                 Modify the status of datacenter.
13:             end
14:         end
15:         if PM$_s$ is empty then
16:             $P_1 = P_1 - 1$.
17:             break.
18:         else
19:             $\rho_e = \rho_e - 1$.
20:         end
21:     end
22:     if $\rho_b = \rho_e$ then
23:         Procedure 1.
24:     else if $\rho_b < \rho_e$ then
25:         Procedure 2.
26:
27: end
28: $P_2 = P_1$.
29: return $P_2$, $S_3$.
```

In order to realize the migration efficiently, we put all used PMs in a list P_l, then we set two pointers ρ_b and ρ_e in the beginning and end position respectively, and the corresponding PMs are called source PM, PM$_s$ and destination PM, PM$_d$ (line 5 in Algorithm 3). After that, we move the VNFRs from PM$_s$ to PM$_t$ (line 6-27 in Algorithm 3). If all VNFRs in PM$_s$ are moved out, PM$_s$ is empty and the number of used PMs decreases (line 16 in Algorithm 3). If not, we decrease ρ_e (line 19 in Algorithm 3), choosing a new PM as PM$_d$ for the VNFRs in PM$_s$. Lines 22-25 are used to check the causes that break the inner *while* loop (lines 8-21 in Algorithm 3).

Procedure 1. Modify ρ_b and ρ_e when $\rho_b = \rho_e$

```
1: Restore the precious status of datacenter.
2: $\delta = \delta + 1$.
3: Set $\rho_e$ to the end of $P_l$.
4: if $\rho_b + \delta \geq \rho_e$ then
5:     break.
6: else
7:     Exchange the PM corresponding to $\rho_b + \delta$ and $\rho_b$.
8: end
```

Procedure 2. Modify ρ_b and ρ_e when $\rho_b < \rho_e$

```
1: $\rho_b = \rho_b + 1$.
2: Set $\rho_e$ to the end of $P_l$.
3: if $\rho_b = \rho_e$ then
4:     break.
5: else
6:     continue.
7: end
```

If $\rho_b = \rho_e$, it implies that all PMs whose resource utilization are higher than that of PM$_s$ have been traversed, and the VNFRs in PM$_s$ cannot be emptied. So we need to restore the previous status of the datacenter to the beginning of this round adjustment (line 1 in Procedure 1). Then we increase the shift value δ in order to choose the next PM$_s$ (line 2 in Procedure 1), and roll back ρ_e to the end of P_l (line 3 in Procedure 1). If $\rho_b + \delta \geq \rho_e$ (line 4 in Procedure 1), it means that each of the used PMs has been treated as PM$_s$ once, and we should stop the adjustment process (line 5 in Procedure 1). Otherwise, we exchange the PMs corresponding to $\rho_b + \delta$ and ρ_b, making the PM corresponding to $\rho_b + \delta$ as the new PM$_s$.

If $\rho_b < \rho_e$, it implies that PM$_s$ is emptied. In this occasion, we need to choose the next PM in list P_l as the new PM$_s$ (lines 1 in Procedure 2). If $\rho_b = \rho_e$ (line 3 in Procedure 2) after the increasing of ρ_b, it means that each of the used PMs has been treated as PM$_s$ once, and we need to stop the adjustment process.

When T-SAT is completed, the mapping of all VNFRs is determined. Then we can place corresponding VNFs and allocate the resources to each VNF according to the demands of VNFRs on it.

5.4 Complexity Analysis

We give a detailed time complexity analysis about T-SAT in this part.

For Algorithm 1, the complexity mainly focuses on the two *while* loops in lines 4-34 and lines 9-33 respectively. For the inner *while* loop (lines 9-33 in Algorithm 1), the time complexity is

$$|\Gamma_c| + (|\Gamma_c| - 1) + \cdots + 1 = |\Gamma_c|(|\Gamma_c| + 1)/2. \tag{21}$$

We assume that each PM can hold φ SFCRs in average and $|\Gamma|/\varphi$ is an integer for the sake of deriving the formulations clearly. Then for the outer *while* loop (lines 4-34 in Algorithm 1), it runs $|\Gamma|/\varphi$ times, and the total time complexity of Algorithm 1 is

$$\begin{aligned}&|\Gamma|(|\Gamma| + 1)/2 + (|\Gamma| - \varphi)((|\Gamma| - \varphi) + 1)/2 + \cdots + \\ &\varphi(\varphi + 1)/2 = (|\Gamma| + \varphi)(2|\Gamma|^2 + \varphi|\Gamma| + 3|\Gamma|)/(12\varphi).\end{aligned} \tag{22}$$

It can be seen that the time complexity of Algorithm 1 is at the level of $\mathbf{O}(|\Gamma|^3/\varphi)$.

For Algorithms 2 and 3, we first introduce two variables, N_c and η, which are the number of the clusters containing used PMs and the number of PMs in one cluster respectively. So there are $N_c \cdot \eta$ used PMs in total at most. The time complexity of the *while* loop (lines 9-19 in Algorithm 2) mainly results from the process of line 10 and line 11 in it. Each PM contains Φ kinds of VNFRs at most, so the time complexity of line 10 in Algorithm 2 is $\eta \cdot \Phi$ at most. The time complexity of line 11 in Algorithm 2 (Function 1) is 5η. So the time complexity of the *while* loop (lines 9-19 in Algorithm 2) is $\eta(\eta\Phi + 5\eta)$, and the time complexity of Algorithm 2 is $N_c\eta(\eta\Phi + 5\eta)$, which is at the level of $\mathbf{O}(N_c\eta^2)$.

Knowing the total number of VNFRs, which is $\sum_{\gamma=0}^{|\Gamma|-1} |\Psi_\gamma|$, and the total used PMs after the first stage, which is $N_c \cdot \eta$. Then in each PM, there are about $\sum_{\gamma=0}^{|\Gamma|-1} |\Psi_\gamma|/(N_c\eta)$ VNFRs in average. So for Algorithm 3, the iteration times of the inner *while* (lines 8-21 in Algorithm 3) is $\sum_{\gamma=0}^{|\Gamma|-1} |\Psi_\gamma|/(N_c\eta) \cdot (N_c\eta) = \sum_{\gamma=0}^{|\Gamma|-1} |\Psi_\gamma|$. We need to traverse all the used PMs to see if it can be emptied. So the above inner *while* loop runs $N_c\eta$ times, then the time complexity upper bound of Algorithm 3 is $N_c\eta \sum_{\gamma=0}^{|\Gamma|-1} |\Psi_\gamma|$. Considering $|\Gamma|/\varphi = N_c\eta$, the time complexity of Algorithm 3 is at the level of $\mathbf{O}(\sum_{\gamma=0}^{|\Gamma|-1} |\Psi_\gamma||\Gamma|/\varphi)$.

In general, the total complexity of correlation-based T-SAT solution is $\mathbf{O}(|\Gamma|^3/\varphi + N_c\eta^2 + \sum_{\gamma=0}^{|\Gamma|-1} |\Psi_\gamma||\Gamma|/\varphi)$. We should notice that η is constant when the datacenter topology is determined, N_c is also limited by the size of datacenter and $\sum_{\gamma=0}^{|\Gamma|-1} |\Psi_\gamma| < |\Gamma|^2$. So we can see that the time complexity of T-SAT is at the level of $\mathbf{O}(|\Gamma|^3/\varphi)$.

6 Performance Evaluation

In this section, we evaluate T-SAT in detail and compare it with the modified FFD (FFD_w), PowerNets [27] and Algorithm H (AH) [9]. In FFD_w, we sort all the VNFRs in descending order based on their average resource demand first, and then the VNFRs are mapped on the PMs one by one based on first-fit (FF) algorithm. For each group of results, we use the average of the results from 10 groups of experiments to reduce the accidental errors.

6.1 Simulation Results with Gaussian Data

6.1.1 Simulation Settings

In evaluation, we use a 3-layer fat-tree of Clos topology [32], where the size of the topology is determined by the number of ports in the switches. For a k-ary Clos topology, there are $(k/2)^2$ k-port core switches; k pods, each of which containing two layers of $k/2$ switches; and $k^3/4$ hosts. We set $k = 16$ in the simulation. Each PM has 100 units CPU resources, 100 units memory resources, and the link capacity is a variable. Both α and β are set to be 0.5 in Function 1. $\theta_{cpu}, \theta_{mem}$ and θ_{link} are all set to be 0.

6.1.2 Workload of SFCRs

The workload of one SFCR is the linear accumulation of all VNFRs' workloads in it. For one VNFR, its CPU, memory and bandwidth workload are all generated according

$$G_x(\tau, \sigma) = \sum_{i=0}^{\kappa-1} \frac{A_i}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(x - \tau_i)^2}{2\sigma_i^2}\right). \tag{23}$$

Eq. (23) is the accumulation of κ Gaussian functions. $\frac{A_i}{\sqrt{2\pi}\sigma_i}$ indicates the amplitude of ith Gaussian function. Because every Gaussian function has only one peak, the value of which equals the amplitude, and κ τ_is are different from each other, so κ also indicates the number of peak values of one workload. In the simulation, we have 50 percent Elephant VNFRs whose CPU A_i, memory A_i and bandwidth A_i are all randomized values obeying uniform distribution from 2 to 3 units and 50 percent Mice VNFRs whose CPU A_i, memory A_i and bandwidth A_i are all randomized values obeying uniform distribution from 0.2 to 0.3 units. For both the Elephant and Mice VNFRs, the σ_is are randomized values obeying uniform distribution from 0.35 to 0.4. For one SFCR, the VNFRs in it belong to only one type, either Elephant VNFR or Mice VNFR.

We assume that the time interval T is a day. As we know, τ_i determines the position of the peak value of one Gaussian function, so we separate the SFCRs into three types according to different τ_is:

- Day SFCR (d-SFCR): The peak loads of this kind of SFCR mainly show up at daytime. We apply the 24-hour clock, so τ_is are chosen from time points [8,9,10,11,14,15,16,17,20].
- Night SFCR (n-SFCR): The peak loads mainly show up at night. And τ_is are chosen from time points [0,1,2,3,4,5,22,23].
- Randomized Time SFCR (r-SFCR): The peak loads show up randomly at the whole day. It occupies a share of the total SFCRs, and the rest are separated evenly between the d-SFCR and the n-SFCR.

Fig. 3 shows the Gaussian workloads of three types of SFCRs, in which the numbers with the underline are the indexes of different kinds of VNFRs, which vary from 0 to 19, so $\Phi = 20$ in the simulation. SFCRs need not to have the same number or same kinds of VNFRs in our simulations. In fact, the number of VNFRs in each SFCR is a random variable varying from 1 to 20. In addition, in Fig. 3, $\mathrm{in\_band}$ is the traffic that flows in one VNFR, while $\mathrm{out\_band}$ is the traffic that flows out of one VNFR. Generally, $\mathrm{out\_band}$ is different from $\mathrm{in\_band}$ after the processing of one VNF. Because $G_x(\tau, \sigma)$ is a continuous function, we apply the uniform sampling to the workload to get the workload vectors. In the simulation, we sample the workload every other 0.1 from 0 to 24.

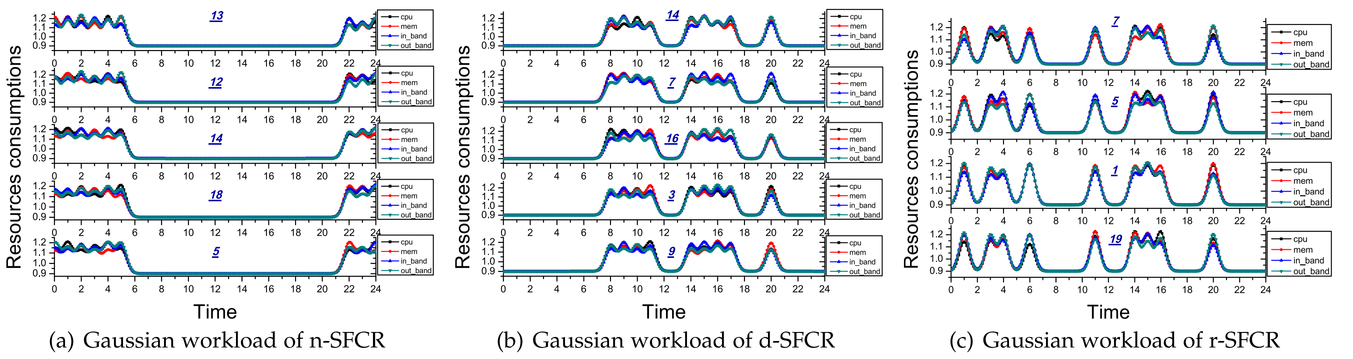
(a) Gaussian workload of n-SFCR (b) Gaussian workload of d-SFCR (c) Gaussian workload of r-SFCR

Fig. 3. Gaussian workload of different kinds of SFCRs.

6.1.3 Results with Gaussian Data

a) Performance Comparisons with Different BRCs. We expose the influence of BRCs on the performance of T-SAT in this part. To demonstrate the relationship more briefly and clearly, we set values of c-BRCs and m-BRCs as the same. Fig. 4a shows the results. Error bars represent the 95 percent confidence intervals, so are they in the following results.

From the figure, we can see that with the increasing of BRCs, T-SAT always performs better than the benchmarks. It is because we combine the VNFRs that demand the same type of VNF together, and then fewer VNF instances are placed, so BRCs are fewer. Moreover, the bigger the volume of BRCs is, the better T-SAT performs. The fewer PMs, the less power consumption and operation cost consequently.

b) Performance Comparisons with Different Link Capacity. Another factor that may influence the performance of our solution is the link capacity. So in this part, we take the link capacity as the variable and compare the performance of different solutions.

From Fig. 4b, we can see that with the increasing of link capacity, the number of used PMs decreases. Because more bandwidth resources lead to more adequate adjustment of T-SAT, then the used PMs are fewer. Moreover, T-SAT performs much better than the benchmarks when the bandwidth resource is scarce. Because T-SAT is inclined to put the whole SFCR in one PM, and then there is less interacting traffic between the PMs. Subsequently, there are fewer link bottlenecks in the network, and the resources of PM can be utilized more adequately. Besides, T-SAT considers the correlation between the workloads of different SFCRs, which leads to a higher utilization of PMs' resources.

c) Performance Comparisons with Different Number of SFCRs. In this part, we evaluate the performance of different solutions via the varying number of SFCRs. As Fig. 4c shows, the number of used PMs increases with more SFCRs. Meanwhile, T-SAT results in the least PMs. This phenomenon indicates that T-SAT has a consistent advantage over the benchmarks with the increasing of SFCRs.

d) Performance Comparisons with Different Ratio of r-SFCR. In this part, we treat the ratio of r-SFCR as the variable and compare the performance of different solutions. From Fig. 4d, we can see that the more r-SFCRs, the more used PMs resulting from the solutions except for FFD_w. Meanwhile, T-SAT still performs the best.

For the performance declining of T-SAT, it is because the more r-SFCR, the fewer SFCRs whose workloads are complementary, and then the resource consumptions are more. As for the performance of FFD_w, it is because FFD_w does not consider the workload correlation between different VNFRs, so it is more likely to put VNFRs whose workloads are less related together when the share of r-SFCR increases, which increases the utilization of the PMs resources. As a result, the number of used PMs decreases with the increasing of r-SFCR ratio for FFD_w.

e) T-SAT versus Optimal Results. We reveal the gap between the performance of T-SAT and the optimal results in this part. With the help of Gurobi [22], the optimal results are derived when the number of SFCRs is small. In the simulation of this part, each SFCR contains 10 VNFRs in average. From Fig. 5, we can see that T-SAT can acquire near effect compared with the optimal results derived by Gurobi when the number of SFCRs is small.

f) The Utilization of Network Resources. Fig. 6 shows the CDF curves of resource utilization resulting from different solutions. The CDF of resource utilization is calculated using

$$CDF(\varrho_0) = \frac{N(\varrho \le \varrho_0)}{R_t}, \tag{24}$$

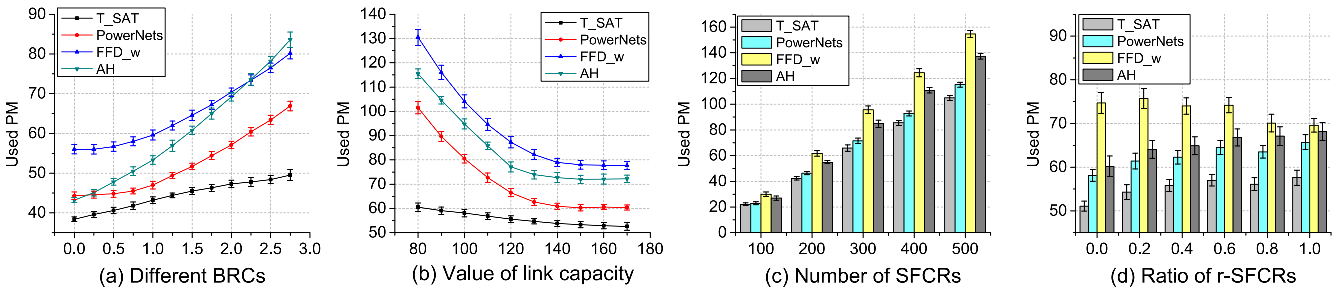
(a) Different BRCs (b) Value of link capacity (c) Number of SFCRs (d) Ratio of r-SFCRs

Fig. 4. Performance comparison of different algorithms with Gaussian data.

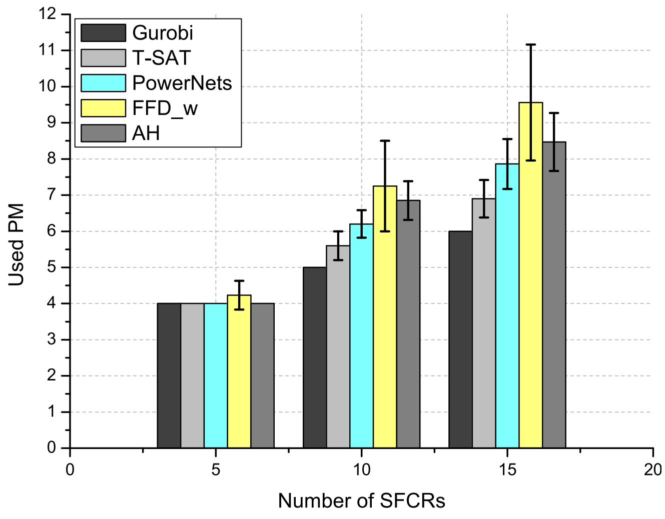

Fig. 5. Performance of different algorithms versus optimal results.

where R_t is the number of total used PMs or links, and $N(\varrho \leq \varrho_0)$ indicates the number of PMs or links whose resource utilization ϱ are below the threshold ϱ_0.

Figs. 6a and 6b show the CDF curves of CPU and memory utilization with BRCs. We can see that the CPU and memory utilization of 80 percent PMs exceeds 70 percent by applying T-SAT, Powernets, and AH, which is very high. The high utilization owes to the consideration of time-varying feature of the workloads. Fig. 6c is the CDF of bandwidth utilization. We can see that the bandwidth utilization is also very high, and the utilization of about 80 percent PMs exceeds 50 percent.

Figs. 6d and 6e are the CDF curves of CPU and memory utilization without accounting for the occupation of BRCs. Excluding BRCs, the resource consumptions are only caused by users' demands. Thus for the same set of SFCRs, the higher resource utilization in the PMs, the fewer PMs will be used. From the figures, we can see that T-SAT has the highest resource utilization excluding BRCs, which strengthens its performance advantages over the benchmarks.

6.2 Simulation Results with Google Data

6.2.1 Simulation Settings

In this section, we evaluate T-SAT with the data derived from Google Cluster Trace [23]. Google cluster trace records the resource utilization of the machines in Google's datacenters in the form of jobs. A job is comprised of one or more tasks, each of which is accompanied by a set of resource requirements used for scheduling (packing) the tasks onto

TABLE 2
Jobs Distribution Based on Task Number

Task Number in a Job	The Corresponding Number of Jobs
1	2,322
2	369
3	230
4	72
5	138
5 to 10	236
10 to 20	126

machines. The relationship between job and tasks is very similar to the relationship between SFCR and VNFRs. So we use jobs in the Google cluster trace to simulate the SFCRs and use tasks in the jobs to simulate the VNFRs in the SFCRs. Table 2 shows the distribution of jobs derived from part of the data set based on the task number in them. We choose the jobs whose task numbers are between 2 and 20 as our simulation data, so the number of total jobs is 1171.

However, the trace only provides the normalized resource consumptions of CPU and memory, so we use Eq. (25) to ca lculate the normalized resource consumptions of bandwidth

$$U_{\text{link}} = 0.5 \cdot U_{\text{cpu}} + 0.5 \cdot U_{\text{mem}}. \tag{25}$$

We use the same topology as that in the Gaussian data part. Because the CPU and memory consumptions are normalized in Google cluster trace, we set both the capacity of CPU and memory resource to 1 unit. As for the link capacity, we treat it as a variable.

6.2.2 Results with Google Data

Fig. 7 shows the results with Google cluster trace data. From the figures, we can draw the consistent conclusions with the Gaussian data. However, we find that the advantages of T-SAT are not so noticeable. This phenomenon exposes one of the shortcomings of correlation-based algorithms, which is sensitive to the data set. When the workloads of different SFCRs or VNFRs have less correlation with each other, the correlation-based algorithms usually can achieve a better performance. However, if the workloads always coincide with each other, the correlation based algorithms do not exhibit their advantages significantly.

Fig. 8 shows the utilization of CPU, memory and bandwidth with Google data. From the figure, we can see that both average utilization of CPU and bandwidth are about 50 percent. But for the memory utilization, it is so high that

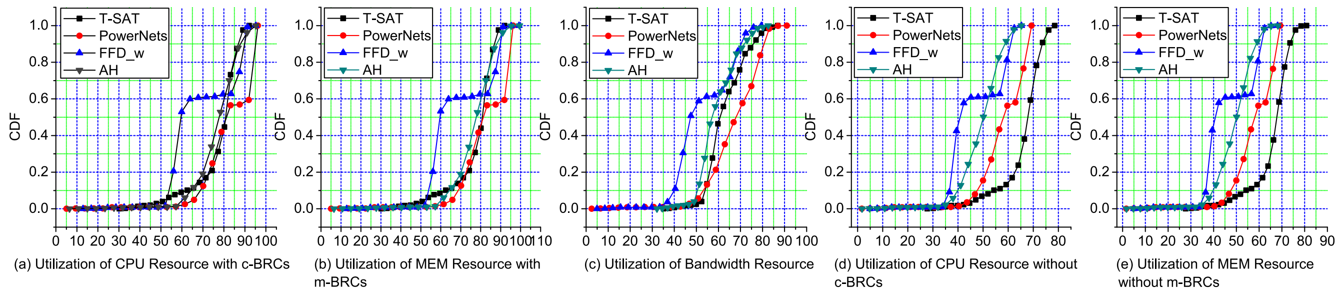

Fig. 6. CDF of network resource utilization with Gaussian data.

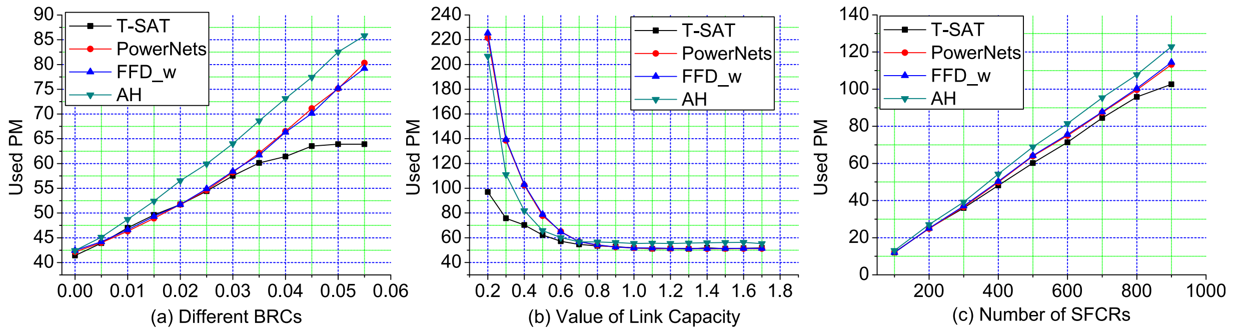

Fig. 7. Performance comparison of different algorithms with Google data.

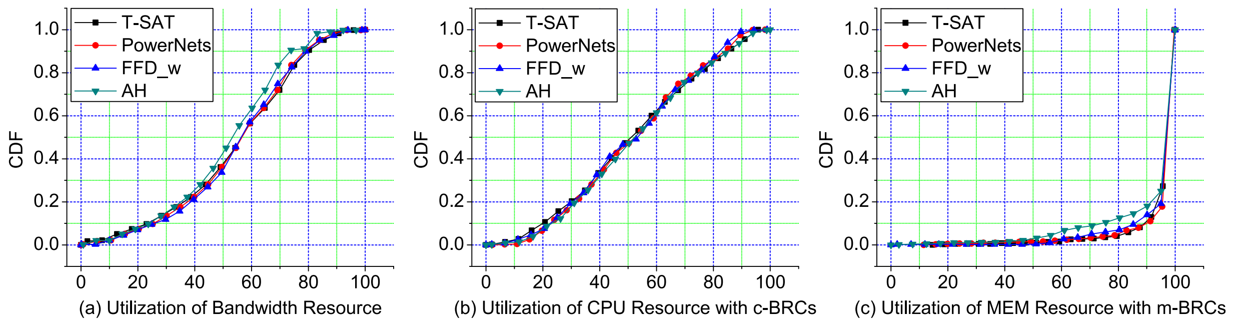

Fig. 8. CDF of network resource utilization with Google data.

about 80 percent PMs exceed 90 percent, from which we can infer that the Google trace data we apply in the simulations has a preference for memory resource.

7 Conclusion

We address the VNF placement problem in cloud datacenter with a set of known SFCRs. In order to optimize the utilization of network resources, the time-varying workloads and BRCs are taken into consideration. Besides, we reveal the confliction between BRCs and the interacting traffic between the PMs. Then we formulate the problem as an ILP model, aiming to minimize the number of used PMs, and propose a two-stage heuristic algorithm, T-SAT. To evaluate the performance of our solution, we make an analysis in detail through simulations with artificial data we compose with Gaussian functions and Google trace data. In the simulations, we compare T-SAT with the benchmarks in existing. The simulation results show that T-SAT can improve the utilization of network resources significantly and acquire a smaller number of used PMs than the benchmarks. Moreover, the performance of T-SAT is very near to the optimal results. Meanwhile, we reveal one shortcoming of the correlation based solutions, which is sensitive to the data set.

In fact, the considerations of time-varying workloads and BRCs not only apply to VNF placement problem in the datacenter networks, and it will be our future work to study the VNF placement problem with the two factors in a wider scenario.

Acknowledgments

The authors sincerely thank the anonymous referees for their valuable suggestions that have led to the present improved version of the original manuscript. This work is partly supported by the National Natural Science Foundation of China under Grant No. 61671420 and No. 61672106, Youth Innovation Promotion Association CAS under Grant No. 2016394 and the Fundamental Research Funds for the Central Universities.

References

[1] Network functions virtualisation (NFV), ETSI, NFVGS, (2015). [Online]. Available: https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf

[2] H. Jeon and B. Lee, "Network service chaining challenges for VNF outsourcing in network function virtualization," in *Proc. Int. Conf. Inf. Commun. Technol. Convergence*, 2015, pp. 819–821.

[3] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, Feb. 2015.

[4] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 13–24, 2012.

[5] G. Gibb, H. Zeng, and N. McKeown, "Outsourcing network functionality," in *Proc. ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2012, pp. 73–78.

[6] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. 3rd Int. Conf. Cloud Netw.*, 2014, pp. 7–13.

[7] 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/Jun. 2016.

[8] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," in *Proc. IEEE INFOCOM*, 2016, pp. 1–9.

[9] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for NFV chaining in packet/optical datacenters," *IEEE J. Lightw. Technol.*, vol. 33, no. 8, pp. 1565–1570, Apr. 2015.

[10] P.-W. Chi, Y.-C. Huang, and C.-L. Lei, "Efficient NFV deployment in data center networks," in *Proc. IEEE Int. Conf. Commun.*, 2015, pp. 5290–5295.

[11] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. IEEE INFOCOM*, 2015, pp. 1346–1354.
[12] J. Halpern and C. Pignataro, "Service function chaining (SFC) architecture," (2015). [Online]. Available: https://rfc-editor.org/rfc/rfc7665.txt
[13] Network Functions Virtualisation (NFV); Management and Orchestration, *NFV-MAN*, ETSI, NFVGS, vol. 1, p. v0, 2014, [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf
[14] C.-P. Bezemer and A. Zaidman, "Multi-tenant SaaS applications: Maintenance dream or nightmare?" in *Proc. Joint ERCIM Workshop Softw. Evolution Int. Workshop Principles Softw. Evolution*, 2010, pp. 88–92.
[15] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, "Design and implementation of a consolidated middlebox architecture," in *Proc. 9th USENIX Conf. Netw. Syst. Des. Implementation*, 2012, pp. 24–24.
[16] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via VM multiplexing," in *Proc. 7th Int. Conf. Autonomic Comput.*, 2010, pp. 11–20.
[17] W. Lin, S. Xu, J. Li, L. Xu, and Z. Peng, "Design and theoretical analysis of virtual machine placement algorithm based on peak workload characteristics," *Soft Comput.*, vol. 21, no. 5, pp. 1301–1314, 2017.
[18] *BEA Weblogic Application Consolidation Strategies*, WLDJ. [Online]. Available: http://weblogic.sys-con.com/node/42938
[19] C.-J. Guo, W. Sun, Z.-B. Jiang, Y. Huang, B. Gao, and Z.-H. Wang, "Study of software as a service support platform for small and medium businesses," in *New Frontiers in Information and Software as Services*. Berlin Germany: Springer, 2011, pp. 1–30.
[20] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *Proc. 11th Int. Conf. Netw. Service Manage.*, 2015, pp. 50–56.
[21] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, Jan.–Mar. 2016.
[22] *Gurobi optimizer reference manual, version 7.0*, Gurobi. [Online]. Available: http://www.gurobi.com/documentation/7.0/refman.pdf
[23] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: Format+ schema," *Google Inc., White Paper*, pp. 1–14, 2011.
[24] W. Rankothge, F. Le, A. Russo, and J. Lobo, "Optimizing resource allocation for virtualized network functions in a cloud center using genetic algorithms," *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 2, pp. 343–356, Jun. 2017.
[25] Z. Á. Mann and A. Metzger, "Optimized cloud deployment of multi-tenant software considering data protection concerns," in *Proc. 17th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, 2017, pp. 609–618.
[26] S. Zhang, Z. Qian, Z. Luo, J. Wu, and S. Lu, "Burstiness-aware resource reservation for server consolidation in computing clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 964–977, Apr. 2016.
[27] K. Zheng, X. Wang, L. Li, and X. Wang, "Joint power optimization of data center network and servers with correlation analysis," in *Proc. IEEE INFOCOM*, 2014, pp. 2598–2606.
[28] D. Banks, J. Erickson, M. Rhodes, and J. S. Erickson, "Multi-tenancy in cloud-based collaboration services," 2009, [Online]. Available: http://www.hpl.hp.com/techreports/2009/HPL-2009-17.pdf?origin=publica tion_detail
[29] S. Natarajan, et al., *An Analysis of Lightweight Virtualization Technologies for NFV*, 2017. [Online]. Available: https://tools.ietf.org/html/draft-natarajan-nfvrg-containers-for-nfv-03
[30] X. Li, A. Ventresque, J. O. Iglesias, and J. Murphy, "Scalable correlation-aware virtual machine consolidation using two-phase clustering," in *Proc. Int. Conf. High Perform. Comput. Simul.*, 2015, pp. 237–245.
[31] V. Perlibakas, "Distance measures for PCA-based face recognition," *Pattern Recognit. Lett.*, vol. 25, no. 6, pp. 711–724, 2004.
[32] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008.

Defang Li received the BS degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 2014. He is working toward the PhD degree at the University of Science and Technology of China with his advisor Peilin Hong now. His research interests include SDN, NFV, and the network resource orchestration and management. He is a student member of the IEEE.

Peilin Hong received the BS and MS degrees from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1983 and 1986, respectively. Currently, she is a professor and advisor for PhD candidates in the Department of EEIS, USTC. Her research interests include next-generation Internet, policy control, IP QoS, and information security. She has published 2 books and more than 100 academic papers in several journals and conference proceedings.

Kaiping Xue (M09-SM15) received the BS degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and the PhD degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. Currently, he is an associate professor in the Department of Information Security and Department of EEIS, USTC. His research interests include next-generation Internet, distributed networks, and network security. He is a senior member of the IEEE.

Jianing Pei received the BS degree from the Department of Information and Electrical Engineering (IEE), China University of Mining and Technology (CUMT), in 2015. He is working toward the MS degree at the University of Science and Technology of China (USTC) with his advisor Peilin Hong now. His research interests include SDN, NFV, and the network resource orchestration and management.