SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

September 2018, Vol. 61 092107:1–092107:19
https://doi.org/10.1007/s11432-017-9357-7

# Virtual network function scheduling via multilayer encoding genetic algorithm with distributed bandwidth allocation

Quan YUAN*, Hongbo TANG, Wei YOU, Xiaolei WANG & Yu ZHAO

*National Digital Switching System Engineering and Technological Research Center, Zhengzhou* 450002*, china*

Received 15 May 2017/Revised 24 September 2017/Accepted 19 December 2017/Published online 13 August 2018

**Abstract** Network function virtualization represents a revolutionary approach to network service deployment. This software-oriented approach for virtual network functions (VNFs) deployment enables more flexible and dynamic network services to meet diversified demands. To minimize the execution time of all VNFs in service function chains, VNF scheduling must be addressed. In this paper, we improve upon the flexible job-shop model by introducing the process of bandwidth allocation. First, we propose a multilayer encoding genetic algorithm to solve the VNF scheduling model. In addition, we design a distributed method for bandwidth allocation based on the Nash bargaining solution. Finally, by combining the genetic algorithm with distributed bandwidth allocation, we present a heuristic algorithm that solves the VNF scheduling problem in one stage. Using a multilayer encoding genetic algorithm, we simplify the constraints of the VNF scheduling problem and reduce its time complexity. At the same time, our Nash game solution refines the granularity of bandwidth allocation to further reduce the transmission delay between VNFs. The effectiveness of our proposed heuristic algorithm is verified through numerical evaluation. Compared with existing approaches, our method exhibits shorter scheduling time and reduces CPU time by 45% in simulated scenarios.

**Keywords** network function virtualization, virtual network function scheduling, genetic algorithm, bandwidth allocation, convex optimization

**Citation** Yuan Q, Tang H B, You W, et al. Virtual network function scheduling via multilayer encoding genetic algorithm with distributed bandwidth allocation. Sci China Inf Sci, 2018, 61(9): 092107, https://doi.org/10.1007/s11432-017-9357-7

## 1 Introduction

Generally, carrier networks are forced to substantially increase both their capital expenditures (CAPEX) and operational expenditures (OPEX) when they deploy or update their network services, as a network service consists of a series of network functions that are implemented by hardware middleboxes (e.g., firewall, load balancers, and intrusion prevention systems). Today, dedicated middlebox hardware is widely deployed in enterprise networks to guarantee network security and performance so that network operators can fulfill their promised service level agreements (SLA). However, the use of middleboxes to provide new services has several inherent shortcomings: (i) Dedicated hardware is always expensive and has a short lifecycle. (ii) The middleboxes of certain network services require a specialized managing panel, which means the hardware of different services cannot be managed or orchestrated via a centralized control plane. (iii) Operators have limited scope for adding new functionalities to existing services or extending the capability of operating systems [1].

* Corresponding author (email: b101180153@smail.nju.edu.cn)

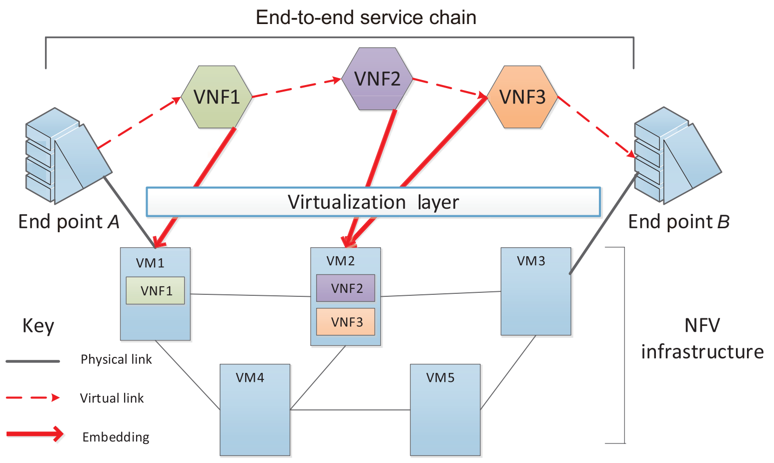

**Figure 1** (Color online) Network service deployment on NFV-enabled platform.

Network function virtualization (NFV) has emerged from the industry and promises to solve the above challenges, thereby avoiding the ongoing proliferation of hardware appliances. NFV enables software-oriented middleboxes to run virtual network functions (VNFs) on commodity servers/virtual machines (VMs). In the NFV ecosystem, a network service is a set of chained VNFs, as illustrated in Figure 1. A network is built and deployed on the NFV platform by (i) defining the number of VNFs to be run, (ii) establishing their execution order in the chain, (iii) embedding the VNFs in the NFV infrastructure, (iv) defining the scheduling strategy inside each VM (inside of VM2 shown in Figure 1). Rather than deploying expensive hardware middleboxes, the implementation of VNFs in software provides more advanced and flexible network services to end users. For a specific network service, the operators can dynamically design a chain of VNFs and steer specific data flows towards that service chain. Based on the volume of data traffic, an efficient VNF deployment policy can be establish to reduce both CAPEX and OPEX [2]. To reduce the occupation time of infrastructure resources, VNF scheduling (VNF-SCH) is a key problem that must be carefully addressed.

VNF-SCH focuses on how to execute each function to minimize the total execution time of the VNFs comprising the network service without degrading service performance while also respecting all the relevant constraints as well as the network function execution order for a particular service. Proper scheduling of VNF execution is necessary to minimize the total execution time of the network services, and thereby reduce the occupation time of infrastructure resources.

Existing approaches for VNF-SCH can be roughly divided into two categories-those that use job-shop model [3–5] and those that rely on heuristics [6, 7]. Job-shop based methods, while theoretically optimal, involve too many constraints in the optimization problem, which can lead to significantly high time complexity. Among the heuristics-based methods, many efficient analytical models have been proposed. However, none can guarantee algorithm performance. Hence, a method with both high algorithm performance and low time complexity is need.

In this research, we designed a dynamic scheduling algorithm for network functions that improves the traditional job-shop model and optimizes bandwidth allocation. The main contributions of this paper are as follows: (i) We propose a multilayer genetic algorithm to solve the VNF scheduling model, which encodes precedence and placement constraints as chromosomes, thereby, enabling the simplification of the optimization problem. (ii) We made correction to the job-shop model according to existing work in the VNF-SCH context, which ignore the optimization of transmission delay. We then design a distributed bandwidth allocation algorithm with fine granularity and low time complexity. (iii) Unlike previous studies, in which the authors solve the VNF scheduling and bandwidth allocation in separated stages, we implement bandwidth allocation during the genetic evaluation iterations. Extensive simulations show that our one-stage method can improve algorithm performance and accelerate iterative convergence.

The rest of the paper is organized as follows. In Section 2, we specify the problem of resource allocation problem in the context of VNF deployment and review existing studies. Section 3 displays the proposed system model and our formulation of VNF scheduling. In Section 4, we propose our solution to VNF scheduling via a multilayer genetic algorithm. In Section 5, we describe our design of a distributed algorithm for bandwidth allocation. In Section 6, we summarize our one-stage algorithm that associates VNF scheduling with bandwidth allocation. In Section 7, we present the numerical results and conclusion is drawn in Section 8.

## 2 Related work

NFV promises to significantly reduce investment cost and improve the efficiency and flexibility of resource allocation. Most recent studies have focused on a compound problem, known as VNF deployment, in which the goal is to satisfy traffic requirements while minimizing the capital and operational costs. In [1], the authors divided VNF deployment into three stages: service function chain (SFC) composition, VNF forwarding graph embedding (VNF-FGE), and VNF-SCH. VNF-SCH is clearly specified as the last stage of VNF deployment stage, and is constrained by the preceding two. The SFC composition process involves determining the precedence constraints that represent the execution order of the VNF. The placement constraints establishing the embedding of VNFs and corresponding VMs are determined after the VNF-FGE. As there is extensive interaction among the three stages, the VNF-SCH problem should not be described separately. In the following, we describe the three stages of VNF deployment.

### 2.1 SFC composition

By utilizing the flexibility of NFV platform, operators can dynamically design SFC topologies for further deployment to achieve particular objectives such as minimizing network delay, maximizing total revenue, and rate of acceptance. Many researches have been proposed to improve the above objectives [8–10]. However, most existing methods consider SFC as a fixed graph wherein all VNFs are executed according to a predefined and fixed sequence. Generally, these predefined VNF sequence comprise the SFC precedence constraints. Obviously, when restricted by fixed precedence constraints, the VNF execution order cannot be dynamically adjusted to achieve the objectives of both VNF-FGE and VHF-SCH, which leads to poorer performances of the two stages. To compose SFCs dynamically, the authors in [11] proposes a flexible SFC composition by slacking the fixed precedence constraint into several precedence dependencies. For example, Figure 2 shows an SFC request with precedence dependencies and three possible VNF chaining options. In Figure 2(a), the precedence dependencies are represented by dotted lines with a unidirectional arrow between the VNFs. As we can see in the figure, the dotted line from VNF3 to VNF1 indicates that VNF3 depends on VNF1 and must therefore be executed after VNF1. Such dependencies are common in real network situations, e.g., the mobile management entity (MME) must be invoked before the home subscriber server (HSS) in the upstream service requests of the evolved packet core network (EPC). Based on these dependencies, valid chaining options regarding SFC requests are derived. Figure 2(b) shows three possible chaining options for the SFC requests shown in Figure 2(a). By offering several possible VNF chaining options for an SFC request, the optimal composition can be selected based on the objective of VNF-SCH to obtain better performance.

### 2.2 VNF-FGE

After composing the SFCs, the objective of the second stage, VNF-FGE, is to determined where to best instantiate the VNFs in the network infruastrucure, when we considering a set of requested SFCs. The VNF-FGE stage determines in which physical nodes the VNFs are instantiated regarding the abovementioned placement constraints. We note that there have been many VNF-FGE studies addressing VNF placement, the goals of which are to minimize resource consumption and enhance reliability [2, 12–14]. Although VNF-FGE is not the main issue adressed in this paper, the result of this stage certianly affects the VNF-SCH solution. In [11], the authors conclude that a stiff VNF-FGE algorithm that embeds a VNF on a fixed VM confines the range of possible VNF placement, thus narrowing the feasible domain

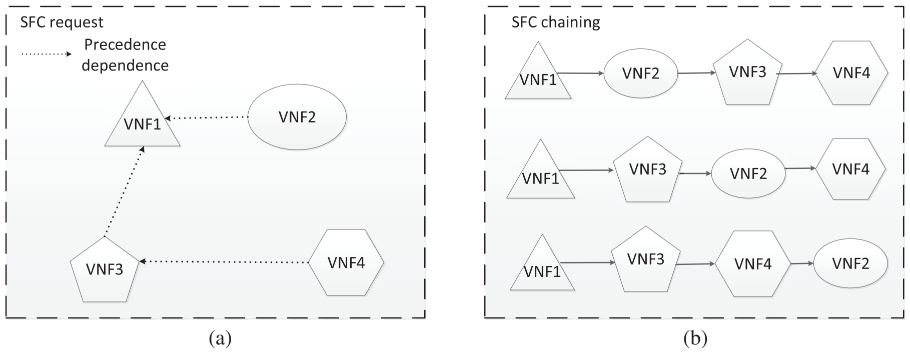

**Figure 2** Flexible composition of SFC. (a) VNF with precedence dependencies; (b) possible chaining options of VNFs.

of VNF-SCH problem. To realize better performance, we slack the placement constraints by predefining a candidate set of VMs for each VNF. The size of each VM set is optionally determined by users.

### 2.3 VNF-SCH

The third and final stage of the VNF deployment problem is the scheduling process, which we denote as VNF-SCH. the objective of this stage attempts to determine how to execute each function while minimizing the total scheduling time, maintaining the service performance, and respecting all the precedence dependencies and placement constraints [1]. Figure 3 shows a simple example on how four different SFC requests can be scheduled over a limited NFV infrastructure, comprising three VMs in this case. It is clear that we can reduce the scheduling time by adjusting the execution order of the VNFs instantiated on the same VM. In this case, if we instantiate the VNF2 of SFC2 prior to the VNF2 of SFC1 during the time slot of $[t_1, t_3)$, the scheduling time would be reduced from $t_5$ to $t_4$.

Existing VNF-SCH studies can be divided into two categories-those that rely on heuristics [6, 7] and those that use the job-shop model [3–5]. For the heuristics-based approaches, an adaptive VNF-SCH model based on mixed integer programming (MIP) was proposed in [6]. Then, the authors in [7] proposed an analytical approach, which designed a greedy algorithm to optimize the execution time of the deployed network service. For methods based on the job-shop model, in [4], the authors first applied a classic job-shop model in the VNF-SCH. They assumed that one VM can only run one VNF at a time slot. This situation exactly matches the case of job-shop problem. In [3], the authors define the VNF scheduling problem as a flexible job-shop problem. By introducing transmission delay into the model, they prove that the optimal solution to the original model can be further optimized. In their scheduling models, both the processing and transmission delay are considered as the optimization objective. In [5], the authors designed an MIP-based scheduling algorithm to reduce the transmissoin delay. Their study divides VNF-SCH into two processes: VNF scheduling and bandwidth allocation. In the first stage, a delay-aware scheduling problem is solved. Then, in the second stage, to minimizing the transmission delay, the bandwidth between VMs is reallocated based on the results of the preceding stage. However, three problems remain unsolved in the above studies. First, the number of constraints involved in the MIP optimization model increases significantly when the scales of network and service increase. Second, the bandwidth allocation method via MIP is characterized by coarse granularity and high time complexity. Third, by executing these two processes separately, the problem of how the bandwidth reallocation impacts the VNF scheduling is neglected, which can affect the convergence efficiency in the iteration process.

To overcome the above limitation, we propose a one-stage multilayer genetic algorithm that concurrently executes bandwidth allocation during the iterative evaluation process. The essence of our algorithm design is as follows. In the genetic evaluation process, we encode precedence and placement constraints into chromosomes, thereby simplifying the scale of the scheduling problem. For the bandwidth allocation process, we designed a distributed algorithm to reduce the time complexity and refine the granularity

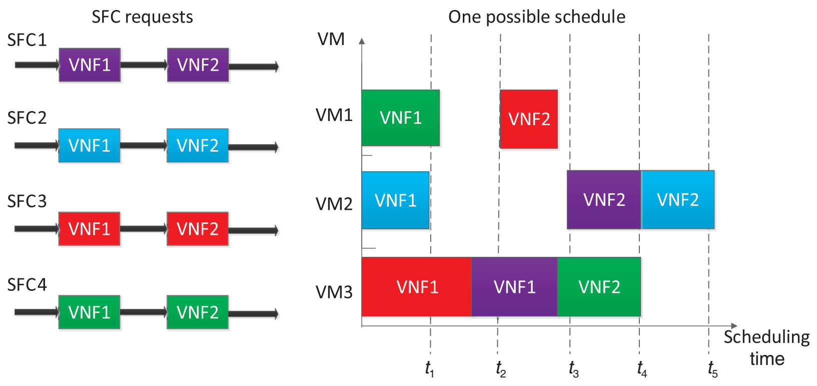

**Figure 3** (Color online) An example of VNF scheduling.

of the allocation procedure. Finally, we combine these two processes into one stage to improve the convergence efficiency of the genetic evaluation.

## 3 Optimization model of VNF-SCH

To conclude the existing methods, fixed constraints are commonly introduced, so the embedding graph and execution order cannot be changed according to the traffic requirements during the VNF-SCH process. In contrast, here, we focus on formulating the VNF-SCH problem by interacting with the other two stages to obtain better performance. In our proposed model, we slack the precedence and placement constraints. With respect to the precedence constraints, we employ a chaining matrix that represents the possible SFC requests with their precedence dependencies rather than a fixed sequence of requested VNFs. For the placement constraints, we use a dynamic candidate VM set, the scale of which is optional and users can opt to enlarge the vector space of the VNF placement. The main objective of VNF-SCH is as follows: given a set of SFCs, each chain having precedence and VNF placement constraints, VNF-SCH identifies the shortest scheduling time for all the service flows. This schedule time is divided into two parts: processing delay and transmission delay. In this paper, we optimize both. With these objectives and constraints, we describe the VNF-SCH problem as follows.

First, we introduce the following formal notations.

- $s$: a given maximum number of SFCs.
- $m$: a given maximum number of VMs.
- $v_T$: a given maximum number of VNF types.
- $v_S$: a given maximum number of VNFs comprising one SFC.
- $v$: a given maximum number of VNFs.
- $P_{v\times m}$: a processing time matrix where each element $P_{i,j}$ denotes the processing time of VNF$i$ running on VM$j$.
- $R_{v\times v}$: an adjacency matrix where each element $R_{i,j}$ denotes the bandwidth allocated to the virtual link from VNF$i$ to VNF$j$.
- $B^{\mathrm{Max}}_{1\times m}$: A bandwidth capacity vector where the element $B^{\mathrm{Max}}_i$ denotes the throughput of VM$i$. We assume that each VM is equipped with a full-duplex Ethernet adapter, so the ingress bandwidth capability of a VM equals that of the egress.
- $F_{s\times v}$: a given bandwidth demand matrix where each row represents the bandwidth demand of an SFC. $F_{i,j}$ denotes the forwarding traffic traversing across VNF$i$ of SFC$j$.
- $C_{s\times v}$: a binary chaining matrix in which rows denote SFCs and columns denote VNFs. $C_{i,j} \in \{0,1\}$, wherein $C_{i,j} = 1$ represents the assignment of VNF$j$ to SFC$i$, otherwise $C_{i,j} = 0$.
- $E_{v\times m}$: a binary placement matrix in which rows denote VNFs and columns denote VMs. $E_{i,j} \in \{0,1\}$ wherein $E_{i,j} = 1$ indicates that VNF$i$ is embedded on VM$j$, otherwise $E_{i,j} = 0$.

• $M_{1\times v}$: a given vector of placement constraints. Each element $M_i$ is a cell denoting the candidate VM set. $M_i$ represents the set of candidate VMs that can instantiate VNF$i$ and $M_i(k)$ is the $i$th element of the cell $M_i$.

• $X_{s\times v}$: a scheduling time matrix wherein the rows denote SFCs and columns denote VNFs. Each element $X_{i,j}$ denotes the time when the traffic traversing across the VM that hosts VNF$j$ of SFC$i$ begins to be processed.

• $Y_{s\times v}$: a scheduling time matrix in which the rows denote SFCs and columns denote VNFs. Each element $Y_{i,j}$ denotes the time when traffic forwarding begins on one of the outgoing virtual links connecting the VM that hosts VNF$j$ of SFC$i$.

Finally, we model the VNF-SCH problem as follows. Minimize

$$\varepsilon = \max\{X_{i,j} + C_{i,j} \cdot\ E_{j,M_j(k)} \cdot P_{j,M_j(k)}\}, \quad i \in [1,s]_Z, \quad j \in [1,v]_Z, \quad k \in [1, \text{length}(M_j)]_Z. \tag{1}$$

Subject to

$$X_{i,j} + C_{i,j} \cdot E_{j,M_j(k)} \cdot P_{j,M_j(k)} \leqslant Y_{i,j}, \quad \forall i \in [1,s]_Z, \quad \forall j \in [1,v]_Z, \quad \forall k \in [1,\text{length}(M_j)]_Z, \tag{2}$$

$$Y_{i,j} + \frac{F_{i,j}}{R_{j,j+h}} \leqslant X_{i,j+h}, h = \min\{\alpha | C_{i,j+\alpha} = 1, \alpha > 0\}, \quad \forall i \in [1,s]_Z, \quad \forall j \in [1,v]_Z, \tag{3}$$

$$\sum_i E_{i,j} \sum_k R_{i,k} \leqslant B_j^{\text{Max}}, \quad \forall i,k \in [1,v]_Z, \quad \forall j \in [1,m]_Z, \tag{4}$$

$$\sum_i E_{i,j} \sum_k R_{k,i} \leqslant B_j^{\text{Max}}, \quad \forall i,k \in [1,v]_Z, \quad \forall j \in [1,m]_Z. \tag{5}$$

Eq. (1) describes the objective of the VNF-SCH, where $\varepsilon$ denotes the time when the last VNF in all the SFCs finishes processing the traffic traversing through it. Eq. (2) ensures that traffic forwarding from a given VNF should not start unless the VNF finishes processing the traffic of the corresponding SFC. Eq. (3) ensures that the traffic processing for a given VNF should not start unless the traffic is received by the VNF from an upstream VNF. $h = \min\{\alpha | C_{i,j+\alpha} = 1, \alpha > 0\}$ ensures that $C_{i,j+h}$ is the first non-zero element on the right of $C_{i,j}$ in row $i$, which implies that VNF$(j+h)$ is the next VNF connected to VNF$j$ in SFC$i$. According to [5], the transmission delay of the virtual link between two VNFs can be simply calculated as $F_{i,j}/R_{j,j+h}$, where $F_{i,j}$ denotes the traffic traversed through VNF$j$ of SFC$i$ and $R_{j,j+h}$ denotes the bandwidth between a corresponding VNF pair. Eqs. (4) and (5) ensure that the sum of the ingress and egress bandwidths allocated to the VNFs instantiated on one VM does not exceed the bandwidth capacity of the VM.

Compared with the MIP model proposed in [3–5], our optimization model does not need to calculate precedence and placement constraints, which are directly encoded into different layers of the chromosome in our multilayer encoding algorithm, as described in the next subsection. Without having to calculate these two types of constraints, our model achieves greater efficiency in obtaining the optimal scheduling time for each feasible case.

## 4 Multilayer encoding algorithm

In the optimization model proposed above, the basic VNF scheduling problem can be viewed as an extension of the classical job-shop problem which is considered to be NP-hard [3]. Joint VNF scheduling with bandwidth allocation is at least as complicated and therefore difficult to solve in polynomial time. As such, we have designed an efficient heuristic approach that uses genetic algorithm to yield a good performance.

### 4.1 Encoding of chromosome and scheduling matrix

In this subsection, we encode a chromosome representing the scheduling order and the embedding of VNFs for all the SFCs as a sequence of integer bits. In each chromosome, bits are divided into two equal-length layers. Bits in the first layer contain the precedence constraints and those in the second layer comprise the placement constraints.

$$\text{chrom}_{\lambda_i} = \{[24311234]\ [21332213]\}. \tag{6}$$

Eq. (6) shows a chromosome example of the scheduling case in Figure 3. As shown above in Figure 3, the chromosome, $\text{chrom}_{\lambda_i}$, consists of 4 SFCs, all of which are composed of two VNFs. All the VNFs are instantiated on 3 VMs. if we consider the first layer of $\text{chrom}_{\lambda_i}$, the execution sequence of all the VNFs is encoded into the first eight bits, and the integer of each bit represents the SFC number and the occurrence number of that integer represents the VNF order in the corresponding SFC. For instance, the first bit 2 indicates that the execution sequence will start with the first VNF of SFC2, whereas the last bit 4 which has occurred for the second time in the sequence represents it will end with the second VNF of SFC4. To represent the first eight bits, we use $\text{VC}_j(i)$ to denote the $i$th VNF of SFC$j$. Accordingly, the first layer of the chromosome, [24311234], is decoded as a possible scheduling sequence, $\text{VC}_2(1)\rightarrow\text{VC}_4(1)\rightarrow\text{VC}_3(1)\rightarrow\text{VC}_1(1)\rightarrow\text{VC}_1(2)\rightarrow\text{VC}_2(2)\rightarrow\text{VC}_3(2)\rightarrow\text{VC}_4(2)$, where $\text{VC}_j(i)$ denotes the $i$th VNF of SFC$j$, such as $\text{VC}_2(1)$ representing the first VNF of SFC2. To simplify these notations, we encode the first eight bits in a scheduling vector $S_{\lambda_i}$ shown in (7), where 201 represents $\text{VC}_2(1)$, the first VNF of SFC2, and the other bits are decoded by such analogy.

In the second layer, the second eight bits, [21332213], represent the corresponding VMs that instantiate the VNFs of the first layer. First, we use $M_{\text{VC}_j(i)}$ to represent the candidate VM cell of VNF $\text{VC}_j(i)$. The integer of each bit represents the element index of the candidate cell. For example, the first bit 2 indicates that we select the second element from the candidate cell of the first VNF of the first layer, namely VM2 from the cell $M_{\text{VC}_2(1)}$=(VM1,VM2,VM3) in this case. Similarly, the other bits can be understood by such analogy and the final scheduling result of the chromosome is shown in Figure 3.

$$S_{\lambda_i} = [201, 401, 301, 101, 102, 202, 302, 402]. \tag{7}$$

In Algorithm 1, we propose our multilayer encoding algorithm. We predefine the size of the population to be NIND which represents the number of chromosomes in the population. First, we sum the rows of the chaining matrix $C_{s\times v}$ to obtain the number of VNFs for each SFC. Then, we encode the SFCs to generate the chromosomes. For lines 6–12, considering the precedence constraints, we generate an SFC and encode it into the bits of the first half chromosome as the first layer, which represents the VNF scheduling order. The order of the VNFs is generated randomly and we examine whether there remains any unscheduled VNFs of corresponding SFC in line 8. For lines 13–16, limited by the placement constraints, we embed the VNF sequence into the corresponding VMs. The VM scheduling order is encoded as bits in the second layer. According to the bits in the first half, we can obtain the corresponding candidate VM cell. Then, we randomly select a VM in the cell and generate the VM scheduling order.

### 4.2 Fitness calculation and ranking

Eq. (8) denotes the fitness of the calculated chromosome as the reciprocal of the objective defined in (1). The $\varepsilon$ of each chromosome can be calculated according to (1)–(5). Then, we use the roulette method for ranking. Eq. (9) indicates the possibility that a chromosome can be selected for further iteration.

$$\text{fitness}(\lambda_i) = \frac{1}{\varepsilon}, \tag{8}$$

$$p_i = \frac{\text{fitness}(\lambda_i)}{\sum_i^{\text{NIND}} \text{fitness}(\lambda_i)}. \tag{9}$$

**Algorithm 1** Multi-layer encoding algorithm (MEA)

**Input:** chaining matrix $C_{s\times v}$ and placement constraints vector $M_{1\times v}$;
**Output:** population chrom$_{\text{NIND}\times l}$ and scheduling matrix $S_{\text{NIND},l/2}$;

```
 1: Generate VNF number vector $N_{s\times 1} = N'_{s\times 1} = \text{sum}(C_{s\times v}, 2)$ ;
 2: $j = 0, i = 0$;
 3: while $j \neq \text{NIND}$ do
 4:     $j+ = 1$;
 5:     while $i \neq \frac{l_i}{2}$ do
 6:         $i+ = 1$;
 7:         $\text{val} = \text{unidrnd}\,(s)$;
 8:         while $N_{\text{val},1} = 0$ do
 9:             $\text{val} = \text{unidrnd}\,(s)$;
10:         end while
11:         $\text{Chrom}_{j,i} = \text{val}$;
12:         $N_{\text{val},1} = N_{\text{val},1} - 1$;
13:         $\text{vnf} = \text{VC}_i(N'_{\text{val},1} - N_{\text{val},1})$;
14:         $\text{VM\_cell} = M_{1,\text{vnf}}$;
15:         $j = \text{unidrnd}\,(\text{size}(M_{1,\text{vnf}}))$;
16:         $\text{chrom}_{j,i+l_i/2} = \text{VM\_cell}(j)$;
17:         $S_{j,i} = 100\text{val} + (N'_{\text{val},1} - N_{\text{val},1})$
18:     end while
19: end while
```

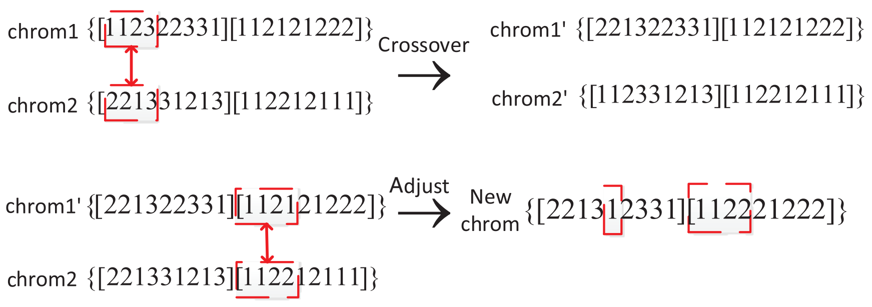
chrom1 {[112322331][112121222]}
chrom2 {[221331213][112212111]}
Crossover
chrom1' {[221322331][112121222]}
chrom2' {[112331213][112212111]}
chrom1' {[221322331][112121222]}
chrom2 {[221331213][112212111]}
Adjust
New chrom
{[221312331][112221222]}

**Figure 4** (Color online) Chromosome crossover process.

## 4.3 Crossover

The population obtains new chromosomes by crossover and then selects chromosomes of high fitness to evolve into the next generation. In our study, we apply the integer crossover method, as illustrated in Figure 4. First, we randomly select two chromosomes and extract the first-half bits. Then, we randomly select a position and cross the two chromosomes. As we can see in Figure 4, chrom1 and chrom2 are two chromosomes in the population, each of which is encoded into 18 bits. Analyzing the first-half bits, we find that there are three SFCs in this case, comprising 3 VNF respectively. Two VMs are provided to host them. At the start of the crossover, we randomly select the fifth bit of chrom1 and chrom2 as a marked position. Next, we cross the first four bits of the two chromosomes to get chrom1′ and chrom2′. After this crossing, two problems emerge for chrom1′. First, a VNF is missing in SFC1 and there is an extra VNF in SFC2. Second, in the first four bits, the order of the VMs does not match that of the VNFs. Therefore, we adjust the first bit with the surplus VNF in SFC2 behind the marked position of chrom1′ (In Figure 4, it is the fifth gene), from 2 into 1. In addition, we cross the genes representing the corresponding tenth to thirtenth VMs, which are supposed to instantiate the VNFs related to the first four bits.

## 4.4 Mutation

To prevent the iteration from falling into local optimum, we apply a mutation operation. First, from the first layer, we randomly select two VNFs on a certain chromosome. Then, we switch the order of the

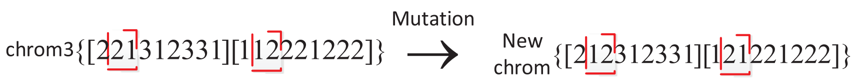
Mutation
chrom3{[221312331][112221222]}
New chrom{[212312331][121221222]}

**Figure 5** (Color online) Chromosome mutation process.

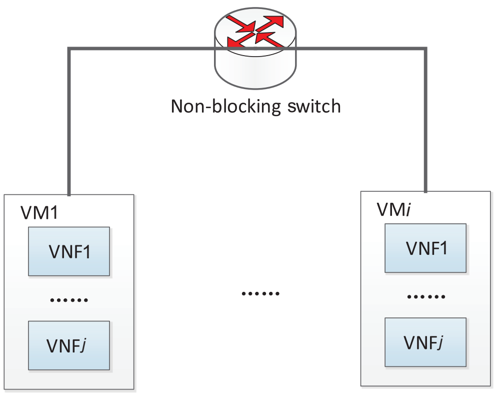

**Figure 6** (Color online) Hose model for VNF bandwidth allocation.

two VNFs and corresponding VMs to generate a new chromosome. Figure 5 shows an example of this mutation. In which, we select two VNFs on the second gene and the third gene. Then we switch the order of the two VNFs and corresponding VMs on the eleventh gene and the twelfth gene to generate a new chromosome.

## 5 Bandwidth allocation algorithm

In most existing studies, researchers applied a classic job-shop model to solve the VNF-SCH problem, ignoring the optimization of the transmission delay. To reduce the transmission delay, a bandwidth allocation scheme must be introduced. The authors in [5] propose a method with bandwidth allocation using MIP. but this method has three shortages: (i) The granularity of this method can be refined, insofar as the software-defined controller directs the data flows traversing through the switches with a granularity of one bit [15]. (ii) By not considering the full-duplex feature of the links between VMs, this method does not distinguish the bandwidth resources in ingress and egress links, so the network cannot be properly depicted when there is an obvious difference between the throughputs of the ingress and egress links. (iii) A scheme for guaranteeing minimum bandwidth is needed to ensure a lower-bound bandwidth allocation independent of the communication patterns of other VNFs. by guaranteeing the bandwidth of each VNF, providers can then negotiate SLAs with respect to network performance with their tenants.

### 5.1 Hose model for VNF bandwidth allocation

To address these shortages, we designed a method using the Nash bargaining solution (NBS). First, we introduce a hose model to illustrate the topology of VM networks as shown in Figure 6 (which was also used in recent proposals [16–18]). In the hose model, VMs are connected to a non-blocking virtual switch and the bandwidth can be fully utilized without considering the topology between the VMs.

In this part, we first initialize the parameters of the hose model for further use. Let $D_{v\times v}$ be the matrix representing the bandwidth demand between VNFs in a datacenter, where $D_{i,j}$ is the bandwidth demand of a VNF pair from VNF$i$ to VNF$j$. To distinguish between the ingress and egress bandwidths of the VNFs (or VMs), we use superscripts I and E, respectively. For example, we denote the total ingress and egress bandwidth demand of VNF$i$ as $D_i^I = \sum_{k=1}^{v} D_{k,i}$ and $D_i^E = \sum_{k=1}^{v} D_{i,k}$, respectively. As defined in Section 3, we can obtain the binary placement matrix $E_{v\times m}$ according to the placement constraints of the chromosomes, where $E_{i,j} = 1$ indicates that VNF$i$ is located on VM$j$. In contrast,

$E_{i,j} = 0$ and $B_i^{\mathrm{Max}}$ indicates the bandwidth capabilities of VM$i$. In our model, we assume each VM be equipped with a full-duplex Ethernet adapter, so the ingress bandwidth capability of a VM equals that of the egress, which are both denoted as $B_i^{\mathrm{Max}}$. According to SLAs, cloud providers must guarantee a minimum quality of service (QoS) for tenants, so a basic bandwidth must be pre-allocated to the virtual link from VNF$i$ to VNF$j$, which is denoted as $B_{i,j}$. The sum of the base bandwidths of all the VNFs hosted on the same VM must always be less maintained than the bandwidth capability of the VM, i.e., $\sum_j^v B_{i,j}E_{i,h} \leqslant B_h^{\mathrm{Max}}$ and $\sum_j^m B_{j,i}E_{i,h} \leqslant B_h^{\mathrm{Max}}$. Finally, we can establish the allocation strategy by detemining the optimal bandwidth allocation matrix $R_{v\times v}$, where $R_{i,j}$ is the bandwidth allocation from VNF$i$ to VNF$j$. Similarly, we can denote the total ingress and egress bandwidth of a VNF as $R_i^I = \sum_{k=1}^v R_{k,i}$ and $R_i^E = \sum_{k=1}^v R_{i,k}$.

## 5.2 Nash bargaining solution

To realize both fair and full utilization, we apply the Nash bargaining solution (NBS) in game theory to solve this bandwidth allocation problem. In the Nash bargaining game, two or more players take part in the game with an initial utility and a utility function. A win-win solution is required, whereby the players cooperate to maximize the social utility, as represented by the Nash product. This exactly mirrors the bandwidth allocation problem in our model, in which VM pairs are guaranteed with a base bandwidth and the operators must maximize their total utility gains with respect to all the VM pairs. As the NBS ensures the Pareto optimality and fairly allocates all resources, we consider NBS to be an appropriate option for solving the bandwidth allocation problem. In [19], the authors were the first to present the Nash bargaining game in the context of the bandwidth allocation problem. The authors in [20] showed that proportional fairness is in fact an NBS. In [21], the authors proposed a theoretical game framework for bandwidth allocation, which not only provides user rate settings that are Pareto optimal for the whole system but is also consistent with the fairness axioms of game theory. In [16], the authors modeled the datacenter bandwidth allocation as a cooperative game, in which guaranteed bandwidth and proportional fairness are considered simultaneously.

Based on previous researches in the field of bandwidth allocation, here, we first present a model of bandwidth allocation in the context of the VNF-SCH problem via NBS. Different from traditional methods, the bandwidth allocation of VNF pairs is aware of traffic demand through its global view of service requests. As such, we designed an online algorithm to achieve fairness in bandwidth allocation and optimizes the transmission delay of VNF pairs.

First, an initial lower bound for the bandwidth of each VNF pair is given in (10). This lower bound ensures that the minimum bandwidth should not exceed the traffic demand can thus be fully used. At the same time, the upper bound is also defined in (11), which ensures that the maximum bandwidth of a VNF pair should not exceed the bandwidth capability of the VM hosting them.

$$L_{i,j} = \min\{D_{i,j}, B_{i,j}\}, \tag{10}$$

$$U_{i,j} = \min\left\{D_{i,j}, \sum_h^m E_{i,h}B_h^{\mathrm{Max}}, \sum_h^m E_{j,h}B_h^{\mathrm{Max}}\right\}. \tag{11}$$

Using $\Re^N$ to represent the vector space of all possible allocations, we denote $X \subseteq \Re^N$ as the vector space of the available solution of bandwidth allocation for $v^2$ VNF pairs, and then use $R_{v\times v} \in X$ to denote the specific allocation result, where the element $R_{i,j}$ denotes the bandwidth allocation of a VNF pair from VNF$i$ to VNF$j$.

According to the definition of the lower bound, a VNF pair whose bandwidth demand is less than the base bandwidth will be allocated a rate that equals to the demand, i.e., $R_{i,j} = D_{i,j}$. Otherwise, this strategy will allocate not only the base bandwidth but also extra bandwidth, which is denoted as $R_{i,j} - L_{i,j}$. We use the extra bandwidth to represent the utility of a VNF pair, i.e. $U_{i,j} = R_{i,j} - L_{i,j}$. In the game, the $v^2$ VNF pairs cooperate with each other to maximize their utility and the Pareto optimization ensures that there is no other allocation that yieldso higher pair utility without sacrificing that of the others.

We use $L_{i,j}$ to represent the initial utility of corresponding VM pair and let $X_0$ be the vector space of the initial utility. The matrix $L_{v\times v} \in X_0$ denotes the initial utility of each VNF pair. As each $R_{i,j}$ has a closed domain, the allocation space $X$ is a convex and closed set [22]. The set of the bandwidth allocation can be defined as $Q = \{R_{v\times v} | R_{v\times v} \in X, R_{i,j} \geqslant L_{i,j}, i,j \in [1,v]_Z\}$, where the bandwidth of each VNF pair is not less than the initial bandwidth. Suppose $Q$ is nonempty and $(Q, L_{v\times v})$ is a bargaining game.

**Definition 1.** A mapping is said to be an NBS if [22]:

(1) $\phi(Q, L_{v\times v}) \in Q$.

(2) $\phi(Q, L_{v\times v})$ is Pareto optimal.

(3) $\phi$ satisfies the linearity axiom if $f: \Re^N \to \Re^N, f(R_{v\times v}) = R'_{v\times v}$ with $R'_{i,j} = a_{i,j}R_{i,j} + b_{i,j}, a_{i,j} > 0, i,j \in [1,v]_Z$, then $\phi(f(Q, L_{v\times v})) = \phi(f(Q), f(L_{v\times v}))$.

(4) $\phi$ satisfies the irrelevant alternatives axiom if $G \subset Q$,$(G, L_{v\times v}) \in (Q, L_{v\times v})$, and $\phi(Q, L_{v\times v}) \in G$ then $\phi(Q, L_{v\times v}) = \phi(G, L_{v\times v})$.

(5) $\phi$ satisfies the symmetry axiom.

According to this definition, we have the following theorem [21].

**Theorem 1.** There exists a Nash bargaining solution and the elements of the set, i.e. solve the following optimization problem.

Let $J$ be the set of VNF pairs that can achieve a utility strictly superior to their initial performance. $J$ is defined as $J = \{R_{v\times v} | R_{v\times v} \in X, R_{i,j} > L_{i,j}, i,j \in [1,v]_Z\}$. The bandwidth allocation of each element in $J$ exceeds the corresponding initial bandwidth. Moreover, the optimization objective can be specified as

$$\max_{R_{i,j}} \prod (R_{i,j} - L_{i,j}), \quad \forall R_{i,j} \in J. \tag{12}$$

Eq. (12) shows the joint profits of the bargaining problem, which is represented by the utility of all the VNF pairs and can be solved by the NBS. To simplify the computing complexity, we convert the optimization objective into logarithmic form. Considering the constraints of each VNF pair, we can get the optimization for THE bandwidth allocation problem ($P_B$) as follows.

Objective:

$$\max_{R_{i,j}} \sum_j \sum_i \ln(R_{i,j} - L_{i,j}), \quad \forall R_{i,j} \in J. \tag{13}$$

Subject to

$$R_{i,j} \geqslant L_{i,j}, \quad \forall i,j \in [1,v]_Z, \tag{14}$$

$$R_{i,j} \leqslant U_{i,j}, \quad \forall i,j \in [1,v]_Z, \tag{15}$$

$$\sum_i R_i^I E_{i,j} \leqslant B_j^{\mathrm{Max}}, \quad \forall i \in [1,v]_Z, \quad \forall j \in [1,m]_Z, \tag{16}$$

$$\sum_i R_i^E E_{i,j} \leqslant B_j^{\mathrm{Max}}, \quad \forall i \in [1,v]_Z, \quad \forall j \in [1,m]_Z. \tag{17}$$

The convex optimization problem has possesses a unique solution that is equivalent to the NBS [21]. Eq. (13) is the logarithmic form of the joint profit of all the players in the bargaining game. Eqs. (14) and (15) ensure that the bandwidth allocation of each VNF pair does not exceed that of the domain. Eqs. (16) and (17) ensure that the sum of ingress and egress bandwidth allocated to the VNFs instantiated on one VM will not exceed the bandwidth capability.

**Proposition 1.** There is a unique NBS $R^*_{v\times v}$ for the centralized optimization problem ($P_B$), where the element $R^*_{i,j}$ denotes the allocation result of the VNF pair from VNF$i$ to VNF$j$. For each element, the solution with Pareto optimization can be characterized as follows.

There exist $\mu_h^I \geqslant 0, h \in [1,m]_Z$ and $\mu_h^E \geqslant 0, h \in [1,m]_Z$ such that for $\forall i,j \in [1,v]_Z$,

$$R^*_{i,j} = L_{i,j} + \frac{1}{\sum_{h=1}^m \mu_h^E E_{i,h} + \sum_{h=1}^m \mu_h^I E_{j,h}}. \tag{18}$$

*Proof.* Now under the assumption that the allocation space is nonempty, convex, and compact, define

$$f(R_{v\times v}) = \sum_j \sum_i \ln(R_{i,j} - L_{i,j}).$$

Then $f(\cdot) : X \to \Re^N$ is strictly concave. We note that the constraints from (14)–(17) are linear, which implies that the first-order Kuhn-Tucker [23] conditions are necessary and sufficient for optimality.

Let $L(R_{v\times v}, \alpha_{v\times v}, \beta_{v\times v}, \mu^I_{1\times v}, \mu^E_{1\times v})$ denote the Lagrangian where $\forall\alpha_{i,j} \geqslant 0$, $\beta_{i,j} \geqslant 0$, $\forall h \in [1,m]_Z$ and $\forall\mu^I_h \geqslant 0$, $\mu^E_h \geqslant 0$, considering the Lagrange multipliers associated with the lower-bound bandwidth in (14), the upper-bound bandwidth in (15), VM bandwidth capacity in (16) and (17), respectively.

Then

$$\begin{aligned} & L(R_{v\times v}, \alpha_{v\times v}, \beta_{v\times v}, \mu^I_{1\times m}, \mu^E_{1\times m}) \\ & = f(R_{v\times v}) + \sum_j^v \sum_i^v \alpha_{i,j}(R_{i,j} - L_{i,j}) - \sum_j^v \sum_i^v \beta_{i,j}(R_{i,j} - U_{i,j}) \\ & \quad - \sum_h^m \mu^I_h((R^I_{1\times v} \cdot E_{v\times m})_h - B^{\mathrm{Max}}_h) - \sum_h^m \mu^E_h((R^E_{1\times v} \cdot E_{v\times m})_h - B^{\mathrm{Max}}_h). \end{aligned} \tag{19}$$

Then the first-order necessary and sufficient conditions are given by $\nabla L(R_{v\times v}, \alpha_{v\times v}, \beta_{v\times v}, \mu^I_{1\times v}, \mu^E_{1\times v}) = 0$. For each VNF pair the Kuhn-Tucker condition is given as follows:

$$\frac{1}{R^*_{i,j} - L_{i,j}} + \alpha_{i,j} - \beta_{i,j} - \sum_h^m \mu^I_h E_{j,h} - \sum_h^m \mu^E_h E_{i,h} = 0$$

and

$$\begin{cases} \alpha_{i,j}(R_{i,j} - L_{i,j}) = 0, & \forall i,j \in [1,v]_Z, \\ \beta_{i,j}(R_{i,j} - U_{i,j}) = 0, & \forall i,j \in [1,v]_Z, \\ \mu^I_h((R^I_{1\times v} \cdot E_{v,m})_h - B^{\mathrm{Max}}_h) = 0, & \forall h \in [1,m]_Z, \\ \mu^E_h((R^E_{1\times v} \cdot E_{v,m})_h - B^{\mathrm{Max}}_h) = 0, & \forall h \in [1,m]_Z. \end{cases} \tag{20}$$

To simplify the optimization problem without loss of generality, we consider the borderlines of constraints (14) and (15) as a special case, whereby we can fully use the bandwidth capability of each VM based on the proportion of the traffic demand and in our method, we handle it separately. Next, we focus on the elements $R_{i,j} \in (L_{i,j}, U_{i,j})$. According to the first two equations in (20), we get $\alpha_{i,j} = \beta_{i,j} = 0, \forall i,j \in [1,v]_Z$. The latter two equations in (20) indicate that the bandwidth capability of all the VMs must be fully utilized, otherwise the multipliers $\mu^I_h$ and $\mu^I_h$ will be forced to zero and no solution can be attained.

Hence the result follows as stated.

Eq. (18) indicates that the optimal bandwidth allocation of the VNF pair from VNF$i$ to VNF$j$ can be simply solved by Lagrange multipliers associated with the VMs hosting the VNFs. As the optimal multipliers of a given VM are independent of each other, we can use a distributed algorithm to solve the centralized optimization problem. Next, we concentrate on attaining the optimal vector of Lagrange multipliers. Given that a VM can host more than one VNF, the number of VNF is usually several times as many VHFs as VMs. Thus, a distributed algorithm can reduce the computational complexity of the convex optimization, where the scale of the vector space of the solution is reduced from $v^2$ VNF pairs to $m$ VMs. We use $\Re^m$ to denote the vector space of all possible solutions of the distributed problem.

The primal problem defined in (13)–(17) can be solved through dual-based decomposition. To normalize the dual function of $P_B$, first we consider an alternative primal problem $P_{B'}$, which has the same optimal solution as $P_B$. The specific objective is given in (21). Based on the conclusion of Proposition 1, we obtain the Lagrangian $L(R_{v\times v}, \mu^I_{1\times m}, \mu^E_{1\times m})$ of problem $P_{B'}$ in (22). Compared to (19), the Lagrangian of centralized problem, Eq. (22) is simplified, where the multipliers $\alpha_{v\times v}$ and $\beta_{v\times v}$ are considered to be zero. To conclude, we get the dual problem $P_D$ corresponding to the primal problem, as described by

(23), where $g(\mu^I_{1\times m}, \mu^E_{1\times m})$ is the dual function and $L(R_{v\times v}, \mu^I_{1\times m}, \mu^E_{1\times m})$ is the Lagrangian.

$$P_{B'} = \min_{R_{i,j}} -\sum_j \sum_i \ln(R_{i,j} - L_{i,j}), \quad \forall R_{i,j} \in J, \tag{21}$$

$$\begin{aligned} L(R_{v\times v}, \mu^I_{1\times m}, \mu^E_{1\times m}) = & -f(R_{v\times v}) + \sum_h^m \mu^I_h ((R^I_{1\times v} \cdot E_{v\times m})_h - B^{\text{Max}}_h) \\ & + \sum_h^m \mu^E_h ((R^E_{1\times v} \cdot E_{v\times m})_h - B^{\text{Max}}_h), \end{aligned} \tag{22}$$

$$\begin{aligned} P_D &= \max_{\mu^I_{1\times m}, \mu^E_{1\times m} \in \Re^m} g(\mu^I_{1\times m}, \mu^E_{1\times m}) \\ &= \max_{\mu^I_{1\times m}, \mu^E_{1\times m} \in \Re^m} \inf_{R_{v\times v} \in \Re^N} L(R_{v\times v}, \mu^I_{1\times m}, \mu^E_{1\times m}) \\ &= \max_{\mu^I_{1\times m}, \mu^E_{1\times m} \in \Re^m} L(R^*_{v\times v}, \mu^I_{1\times m}, \mu^E_{1\times m}). \end{aligned} \tag{23}$$

Since the optimal solution of the primal problem $P_{B'}$ is unique, the corresponding solutions of problem $P_D$ converge to a unique optimal bandwidth allocation matrix $R^*_{v\times v}$. To solve the primal problem $P_{B'}$, we first obtain the optimal solution to the dual problem. Using a constant step, we design a gradient-descent algorithm based on that in [21].

For each $h \in [1, m]_Z$, the recursive equation of $\mu^I_k$ and $\mu^E_k$ is given in (24) and (25), respectively.

$$\begin{aligned} (\mu^I_h)^{k+1} &= \max\left\{0, \left[(\mu^I_h)^{(k)} + \gamma \frac{\partial g(\mu^I_{1\times m}, \mu^E_{1\times m})}{\partial \mu^I_h}\right]\right\} \\ &= \max\left\{0, \left[(\mu^I_h)^{(k)} + \gamma \left[\sum_j^v \left(\sum_i^v R^*_{i,j}\right) E_{j,h}\right] - B^{\text{Max}}_h\right]\right\}, \end{aligned} \tag{24}$$

The bandwidth allocation algorithm is given as Algorithm 2. In lines 2–6, we handle VNF pairs whose bandwidth demand have less priority than their base bandwidth. Next, for further recursion, we initialize the parameters of the Nash bargaining game. On line 11, we update the Lagrange multipliers using a gradient-descent method. Then, we update the corresponding allocation matrix according to Proposition 1 one line 13. In lines 14–17, we detect whether the solution exceeds the upper bound of the bandwidth allocation matrix. Finally, a stop condition is executed to accelerate the recursion. The threshold value of the recursion $\Delta$ is set to a number small enough to guarantee the accuracy of the bandwidth allocation.

$$(\mu^E_h)^{k+1} = \max\left\{0, \left[(\mu^I_h)^{(k)} + \gamma \left[\sum_i^v \left(\sum_j^v R^*_{i,j}\right) E_{i,h}\right] - B^{\text{Max}}_h\right]\right\}. \tag{25}$$

## 6 Multilayer encoding genetic algorithm with bandwidth allocation

In this section, we combine the multi-layer genetic algorithm with the bandwidth allocation solution in one stage. The main reason why existing method cannot execute the process of VNF scheduling and bandwidth allocation in the same stage is that the scheduling models of those methods involve the calculation of precedence and placement constraints, both of which are necessary for the bandwidth allocation solution. As such, VNF scheduling must be solved prior to starting bandwidth allocation. As we have encoded the two constraints into chromosomes, the process of SFC composition and VNF-FGE could be separated from the scheduling model, and we can therefore complete the bandwidth allocation during the iterative evaluation of the multi-layer genetic algorithm.

**Algorithm 2** Bandwidth allocation algorithm

```
Input: Placement matrix $E_{s\times v}$, demand matrix $D_{v\times v}$, base bandwidth matrix $B_{v\times v}$, bandwidth capability matrix $B^{\text{Max}}_{1\times m}$,
    the maximum number of iteration MAXNUM;
Output: The solution of bandwidth allocation $R^*_{v\times v}$;
 1: Initialize the lower bound matrix of bandwidth allocation $L_{v\times v}$ and the upper bound matrix $U_{v\times v}$;
 2: while $i,j \in [1,v]_Z$ do
 3:     if $D_{i,j} \leqslant B_{i,j}$ then
 4:         $R^*_{i,j} = D_{i,j}$;
 5:     end if
 6: end while
 7: Initialize the step-size $\gamma$ and the Lagrange multiplier $(\mu^I_{1\times m})^{(0)}$ and $(\mu^E_{1\times m})^{(0)}$;
 8: Initialize $k = 0$ and $R^{(0)}_{v\times v}$ according to (18);
 9: while $k <$ MAXNUM do
10:     $k = k + 1$;
11:     Calculate $(\mu^I_{1\times m})^{(k)}$ and $(\mu^E_{1\times m})^{(k)}$ according to (24) and (25);
12:     while $R_{i,j} \in J$ do
13:         Calculate $R^{(k)}_{i,j}$ according to (18);
14:         if $R^{(k)}_{i,j} \geqslant U_{i,j}$ then
15:             $R^{(k)}_{i,j} = U_{i,j}$;
16:         end if
17:     end while
18:     if $f(R^{(k+1)}_{v\times v}) - f(R^{(k)}_{v\times v}) \leqslant \Delta$ then
19:         $R^*_{i,j} = R^{(k)}_{i,j}$;
20:         break;
21:     end if
22: end while
```

In Algorithm 3, we propose the entire multilayer encoding genetic algorithm with bandwidth allocation, ME-GA. For lines 1–3, we initialize the network and genetic parameters for further use. In line 4, we execute the MEA algorithm to obtain the initial chromosomes. Then, we solve the bandwidth allocation problem in line 5. For line 6, according to the VNF scheduling model, we calculate the scheduling time matrix of the initial chromosomes. In lines 7–19, we conduct the evolution procedure to find the optimal solution of all chromosomes.

**Algorithm 3** Multi-layer encoding genetic algorithm (ME-GA)

```
Input: Service requests of VNFs with precedence constraints and placement constraints;
Output: The optimal chromosome chrom$_\lambda$ and scheduling time matrix $X_{s\times v}$ and $Y_{s\times v}$;
 1: Compose the SFCs with all possible chaining and generate the parameter $C_{s\times v}$ and $M_{1\times v}$, according to the precedence
    constraints and placement constraints;
 2: Initialize the basic network parameter $P_{v\times m}$, $B_{m\times m}$ and $F_{s\times v}$;
 3: Initialize genetic algorithm parameter;
 4: Encoding SFCs with MEA to attain the population chrom$_{\text{NIND}\times l}$, the placement matrix $E_{v\times m}$ and the scheduling
    matrix $S_{\text{NIND},l/2}$;
 5: Execute the bandwidth Allocate algorithm for each chromosome according to traffic demand;
 6: calculate the scheduling time matrix $X_{s\times v}$ and $Y_{s\times v}$;
 7: while gen < MAXGEN do
 8:     Calculate the fitness and ranking possibility of each chromosome;
 9:     Operate the crossover process and generate new chromosomes;
10:     Operate the mutation process and generate new chromosomes;
11:     Execute the bandwidth Allocate algorithm for new chromosomes;
12:     Calculate the scheduling time matrix $X_{s\times v}$ and$Y_{s\times v}$of new chromosomes;
13:     Calculate the fitness of new chromosomes and ranking current population;
14:     Update current population with high ranking chromosomes;
15:     Select the optimal chromosome for further iteration;
16:     gen = gen+1;
17:     Decode the optimal chromosome chrom$_\lambda$ and save the optimal scheduling matrix $S_{\text{NIND},l/2}$ and scheduling time
        matrix $X_{s\times v}$ and $Y_{s\times v}$;
18: end while
19: Output the optimal chromosome chrom$_\lambda$ and scheduling time matrix $X_{s\times v}$ and $Y_{s\times v}$;
```

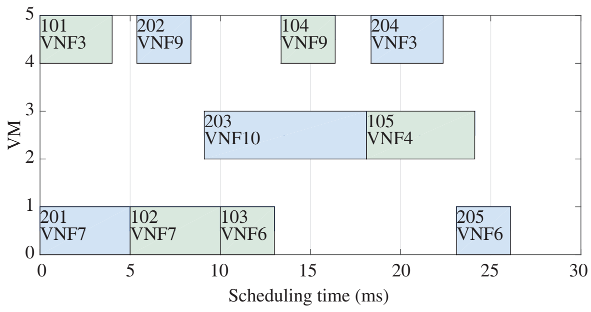

**Figure 7** (Color online) Scheduling result of ME-GA (BRV-GA).

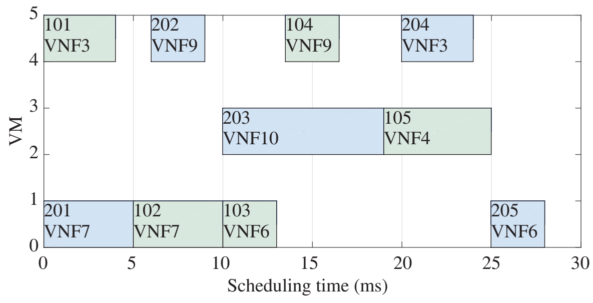

**Figure 8** (Color online) Scheduling result of UB-MIP.

## 7 Simulation results

In this section, we evaluate the performance of our proposed VNF-SCH method (ME-GA) with bandwidth allocation. In comparison, we considered the MIP method with uniform bandwidth allocation (UB-MIP) [4] and the two-stage scheduling method (BRV-GA) proposed in [5].

### 7.1 Parameter configuration

A computer with 4 GB internal storage memory and Intel i7 4790 2.8 GHz CPU model is employed. First, considering the scale of service request and physical network. Referring to the simulation environment in [5], we set the maximum number of SFCs $s = 100$, the maximum number of VMs $m = 20$, the maximum number of types of VNFs $v_T = 10$, and the maximum number of VNF in one SFC $v_S = 5$. Based on the experimental data in [24], we set up our network latency parameter. The elements of the processing time matrix $P_{v\times m}$ are generated randomly in an area of $[2, 10]$ ms. We uniformly initiated the base bandwidth of the links between VNFs to 2 Mbps and set the total throughput of a VM is set to 20 Gbps. Here, we consider two situations in the bandwidth demand matrix $F_{s\times v}$. For requests of control plane, as the traffic is light, the element of $F_{s\times v}$ is set to an area of $(0, 2]$ Mbps, while that is set to $[10, 20]$ Mbps for the data plane. The element of the placement constraints cell vector $M_{1\times v}$ is generated randomly with a domain of $[1, m]_Z$ and the size of each candidate VM cell is restricted to 3 at most, which accords with the majority of VNF-FGE researches. In the recursion of the gradient-descent algorithm, we use the total recursive step MAXNUM = 20, and step-size $\gamma = 0.7$, and set the value of the stop condition $\Delta$ to 0.01. For the parameters of genetic algorithm, the size of population NIND = 40 and the maximum generation of iterations MAXGEN = 50. The possibilities of crossover and mutation are set to 90% and 60% respectively.

### 7.2 Numerical results

First, using a fully connected network with five VMs and two control-plane SFCs comprising five VNFs each, we set MAXGEN = 50 and simulated the performance of the above three methods (UB-MIP, BRV-GA, ME-GA). Figure 7 shows the scheduling results of ME-GA, the strategy of which is the same as that for BRV-GA in this case. Figure 8 shows the scheduling result of UB-MIP, the only method that ignores the bandwidth allocation. In Figures 7 and 8, the $Y$ label denotes the VM number and $X$ label indicates the scheduling time. Blocks of different colors denote the VNFs for different SFCs. In the blocks, the text of the first row indicates the SFC number and scheduling order of the corresponding VNF, where 101 denotes the first VNF of SFC1, and that of the second row illustrates the type of VNF. As control plane traffic is relatively light, the impact of transmission delay is not obvious, while with the process of bandwidth allocation, the two methods apply the same scheduling strategy. In this case, the ME-GA and BRV-GA can both obtain the optimal result of bandwidth allocation to shorten the transmission delay according to the traffic.

**Table 1** Comparison of different methods in control plane

| Method | Scheduling time (ms) | CPU time (s) | Optimality gap (%) |
|---|---|---|---|
| UB-MIP | 28 | 6.3 | 0 |
| BRV-GA | 26.1 | 1.31 | 6.8 |
| ME-GA | 26.1 | 0.98 | 6.8 |

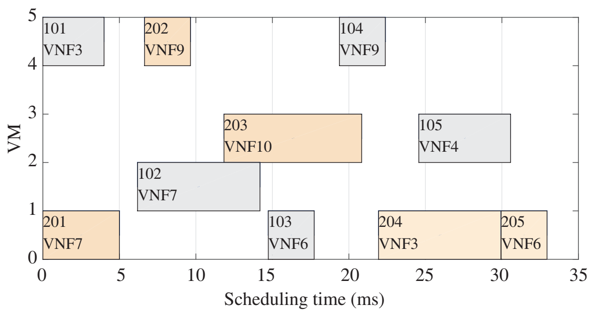

**Figure 9** (Color online) Scheduling result of ME-GA.

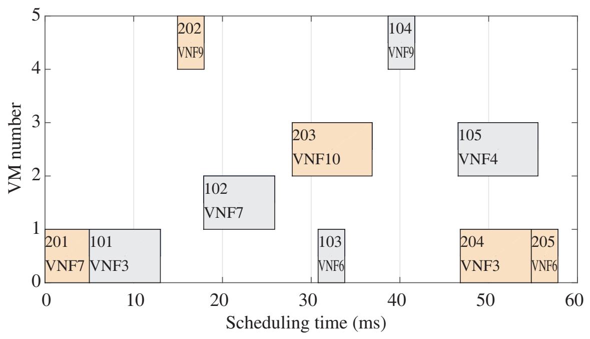

**Figure 10** (Color online) Scheduling result of BRV-GA.

Table 1 shows the comparison of the three methods in terms of CPU and scheduling time. It is clear that UB-MIP has the longest scheduling time, reaching 28 ms without optimizing the transmission delay. Considering bandwidth allocation, the other 2 methods both optimize the scheduling time 6.8% faster. As for CPU time, UB-MIP possesses the longest CPU time for two main reasons: (i) It solves the MIP model by the traditional tool, CPLEX, which can figure out the exact solution of our model. However, the toolbox cannot approach the optimum with a particular gradient, which is usually provided by an intelligent algorithms like the genetic algorithm. (ii) Not considering the process of bandwidth allocation, there may be several possible solutions that obtain the optimal results, which increases the time complexity of the search process. Compared to the performance of the BRV-GA with respect to time complexity, the ME-GA is a slightly superior. Given that the maximum number of iterations is set to the same constant, we can conclude that the ME-GA can provide a more efficient solution of VNF-SCH problem than BRV-GA, in terms of the single loop of iterations. As we have already encoded the precedence and placement constraints into the chromosome, the bandwidth allocation model no longer needs to take them into consideration, which could simplify the optimization constraints optimization and saves more time in calculating the feasible chromosomes of the entire population during the evaluation process. Moreover, the distributed bandwidth allocation algorithm also shows higher efficiency than that of the centralized MIP method, which will be addressed in detail later.

Using the same network topology of VMs, we applied the method of ME-GA and BRV-GA to two data-plane SFCs composed of 5 VNFs each. Figures 9 and 10 show the scheduling results for the ME-GA and BRV-GA, respectively. As the traffic of data plane grows relatively heavy, the impact of transmission delay becomes significant. Due to the fine granularity of the bandwidth allocation, the ME-GA has a shorter scheduling time than the BRV-GA. Comparing Figures 9 and 10, we can draw a conclusion that the result of bandwidth allocation can affect the VNF schedule strategy. By executing VNF schedule and bandwidth allocation as separate stages, the BRV-GA neglects the interaction of these two processes, which may cause it fail to find the optimal policy within limited number of iterations. In the following analysis, we explain the reason for this.

Table 2 shows the comparison in terms of CPU and scheduling time for the three methods in data-plane SFCs. In this case, the transmission delay turns to be the determinant of scheduling time. Reaching 57.8 ms, the poor performance of the UB-MIP indicates the importance of bandwidth allocation. Associated with a method of bandwidth allocation via MIP, BRV-GA improves the performance with an optimization gap of 32.3%. By refining the granularity of the bandwidth allocation, the ME-GA obtains the shortest scheduling time at 33 ms. Furthermore, when we compare Table 1 with Table 2, it is con-

**Table 2** The comparison of different methods in data plane

| Method | Scheduling time (ms) | CPU time (s) | Optimality gap (%) |
|---|---|---|---|
| UB-MIP | 57.8 | 6.3 | 0 |
| BRV-GA | 39.1 | 1.31 | 32.3 |
| ME-GA | 33 | 0.98 | 42.9 |

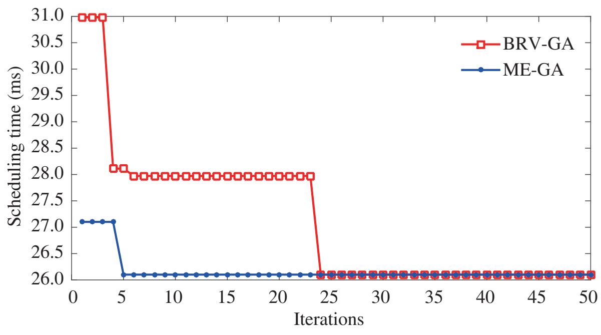

**Figure 11** (Color online) Convergence procedure of scheduling time in control plane.

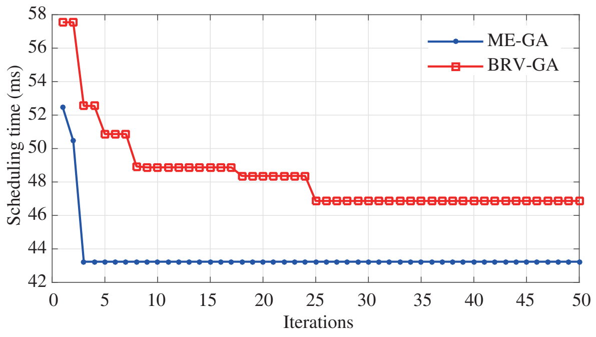

**Figure 12** (Color online) Convergence procedure of scheduling time in data plane.

cluded that the time complexity of all the above method is related only to the scale of the VM network and SFCs.

Figures 11 and 12 illustrate the variation of the scheduling time of the optimal chromosome with the number of iterations for the case of control plane and data plane respectively. As we analyze the result shown in Figure 11, the curve of ME-GA approaches directly to the optimal solution after several iterations, whereas that of BRV-GA suffers a long-term stagnation before it finds the optimum allocation policy. The main reason accounting for the efficient convergence of the ME-GA is that we implement the bandwidth allocation algorithm during the evaluation. Without considering the interaction between bandwidth reallocation and VNF scheduling, the BRV-GA executes the two processes in separate stages. As a result, the optimal chromosome of the first stage (VNF-scheduling) may be replaced by other chromosomes in the second stage (bandwidth allocation). As we can see in Figures 11 and 12, the alternate replacement of optimal chromosomes may lead to the stagnation of convergence and the procedure may ultimately fall into a local optimum. In Figure 12, the curve of the ME-GA shows a plummeting trend and reaches the convergence point rapidly. However, experiencing several stagnations during the iteration, the BRV-GA exhibits the trend of falling into a local optimum. Moreover, compared to the method via MIP, our bandwidth allocation algorithm provides a fine-granularity means of solving the optimization problem. As the traffic grows heavier, the difference of performance between the two methods tends to be more obvious.

Next, we study the effect of different sizes of the network service. Figure 13 depicts the CPU time of the ME-GA and BRV-GA with variation in the number of SFCs. We implemented the algorithms in a large fully connected 20-node virtual network and set the scale of each SFC to no more than five VNFs. In this simulation, we configured the traffic of the network service chains from 0 to 20 Mb uniformly, which can represent the service requests in both the control and data planes. In Figure 13, our method shows superiority in the performance of CPU time, namely, the execution time. In total, the CPU time of the two methods show a linear correlation with the number of SFCs, such that as the number of SFCs increases, the CPU time of the BRV-GA increases more quickly than that of our method. At the starting point of 10 SFCs, the time difference of the two methods is only 0.8593 s, but it increases steadily to 7.7343 s by the end point of 100 SFCs. Moreover, after further calculation, we find that the CPU time of the ME-GA, in general, keeps 45% lower than that of the BRV-GA. Two reasons are listed to explain the low time complexity of our method. First, the multi-layer encoding algorithm encodes the precedence and placement constraints into chromosomes and builds a connection

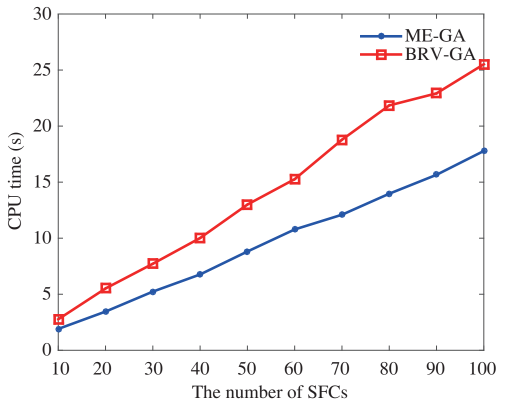

**Figure 13** (Color online) CPU time for different number of SFCs.

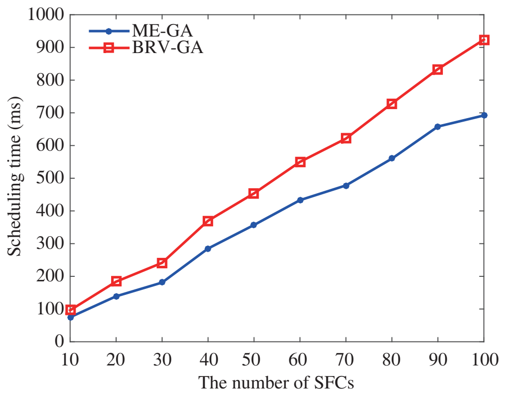

**Figure 14** (Color online) Scheduling time for different number of SFCs.

between the two, so there is no need to calculate them in the optimization problem. Secondly, we apply a distributed algorithm to solve the bandwidth allocation problem, which can shrink the vector space of feasible solutions and improve the time efficiency. Using the same parameter configuration, Figure 14 shows the ascending trend of scheduling time when the number of SFCs increases from 10 to 100. As the SFCs are generated uniformly, the number of them could indicate the traffic traversing across the network of VMs. In total, the performance of our method is better. Furthermore, its superiority is more apparent when the scale of network service increases. As the traffic demand becomes heavier, our fine-granularity bandwidth allocation algorithm clearly shows its advantages in the optimization of transmission delay. In addition, due to the frequent stagnation in the evaluation process, the BRV-GA may fall into a local optimal solution within limited number of iterations.

## 8 Conclusion

In this paper, we studied the important problem of VNF-SCH with transmission delay optimization and proposed a heuristic algorithm via multilayer genetic algorithm with a distributed bandwidth allocation. First, to reduce time complexity, we simplified the optimization problem based on the flexible job-shop model by designing a multi-layer encoding genetic algorithm. Then, we proposed a fine-granularity and distributed bandwidth allocation approach via Nash bargain solution to optimize the transmission delay between VMs. Finally, to accelerate the iterative convergence, we integrated the GA-based algorithm with the proposed bandwidth allocation approach to solve the VNF-SCH problem in just one stage. The numerical simulation results showed that using our one-stage method, we can realize both shorter schedules and lower time complexity. Achieving a shorter schedule enables service providers to admit and serve more flows in its cloud data center, thereby increasing their revenues. In addition, lowering the time complexity can improve the instantaneity of our dynamic algorithm as well as utilization of computational resources.

**Acknowledgements** This work was supported by Foundation for Innovative Research Groups of National Natural Science Foundation of China (Grant No. 61521003), and National Key Research and Development Program of China (Grant No. 2016YFB0801605).

## References

1 Herrera J G, Botero J F. Resource allocation in NFV: a comprehensive survey. IEEE Trans Netw Serv Manage, 2016, 13: 518–532
2 Cao J, Zhang Y, An W, et al. VNF-FG design and VNF placement for 5G mobile networks. Sci China Inf Sci, 2017, 60: 040302

3 Riera J F, Escalona E, Batalle J, et al. Virtual network function scheduling: concept and challenges. In: Proceedings of International Conference on Smart Communications in Network Technologies, Vilanova i la Geltru, 2014. 1–5
4 Mijumbi R, Serrat J, Gorricho J L, et al. Design and evaluation of algorithms for mapping and scheduling of virtual network functions. In: Proceedings of IEEE Conference on Network Softwarization, London, 2015. 1–9
5 AL-Dhahir N. Editorial a message from the new editor-in-chief. IEEE Trans Commun, 2016, 64: 1
6 Riera J F, Hesselbach X, Escalona E, et al. On the complex scheduling formulation of virtual network functions over optical networks. In: Proceedings of International Conference on Transparent Optical Networks, Graz, 2014. 1–5
7 Riera J F, Hesselbach X, Zotkiewicz M, et al. Modelling the NFV forwarding graph for an optimal network service deployment. In: Proceedings of International Conference on Transparent Optical Networks, Budapest, 2015. 1–4
8 Kong Z, Xu C Z, Guo M. Mechanism design for stochastic virtual resource allocation in non-cooperative cloud systems. In: Proceedings of IEEE International Conference on Cloud Computing, Washington, 2011. 614–621
9 Bari M F, Boutaba R, Esteves R, et al. Data center network virtualization: a survey. IEEE Commun Surv Tut, 2013, 15: 909–928
10 Correa E S, Fletscher L A, Botero J F. Virtual data center embedding: a survey. IEEE Latin Am Trans, 2015, 13: 1661–1670
11 Beck M T, Botero J F. Coordinated allocation of service function chains. In: Proceedings of 2015 IEEE Global Communications Conference (GLOBECOM), San Diego, 2015. 1–6
12 Bari M F, Chowdhury S R, Ahmed R, et al. On orchestrating virtual network functions. In: Proceedings of International Conference on Network and Service Management, Barcelona, 2015. 50–56
13 Luizelli M C, Bays L R, Buriol L S, et al. Piecing together the NFV provisioning puzzle: efficient placement and chaining of virtual network functions. In: Proceedings of IFIP International Symposium on Integrated Network Management, Ottawa, 2015. 98–106
14 Moens H, Turck F D. VNF-P: a model for efficient placement of virtualized network functions. In: Proceedings of International Conference on Network and Service Management, Rio de Janeiro, 2014. 418–423
15 Kim H, Feamster N. Improving network management with software defined networking. IEEE Commun Mag, 2013, 51: 114–119
16 Guo J, Liu F, Lui J C S, et al. Fair network bandwidth allocation in IaaS datacenters via a cooperative game approach. IEEE/ACM Trans Netw, 2015, 24: 873–886
17 Ballani H, Costa P, Karagiannis T, et al. Towards predictable datacenter networks. In: Proceedings of ACM SIGCOMM 2011 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Toronto, 2011. 242–253
18 Guo J, Liu F, Zeng D, et al. A cooperative game based allocation for sharing data center networks. In: Proceedings IEEE INFOCOM, Turin, 2013. 2139–2147
19 Mazumdar R, Mason L G, Douligeris C. Fairness in network optimal flow control: optimality of product forms. IEEE Trans Commun, 1991, 39: 775–782
20 Touati C, Altman E, Galtier J. Generalized Nash bargaining solution for bandwidth allocation. Comput Netw, 2006, 50: 3242–3263
21 Yaiche H, Mazumdar R R, Rosenberg C. A game theoretic framework for bandwidth allocation and pricing in broadband networks. IEEE/ACM Trans Netw, 2000, 8: 667–678
22 Nisan N, Papadimitriou C H. Algorithmic game theory. Commun ACM, 1950, 53: 78–86
23 Boyd S, Vandenberghe L. Convex Optimization. Cambridge: Cambridge University Press, 2004
24 Martini B, Paganelli F, Cappanera P, et al. Latency-aware composition of virtual functions in 5G. In: Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), London, 2015. 1–6