SCIENCE CHINA
Information Sciences

• **RESEARCH PAPER** •
Special Focus on Mobile Network Virtualization

April 2017, Vol. 60 040302:1–040302:15
doi: 10.1007/s11432-016-9031-x

VNF-FG design and VNF placement for 5G mobile networks

Jiuyue CAO, Yan ZHANG, Wei AN*, Xin CHEN, Jiyan SUN & Yanni HAN

State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093*, China*

Received November 24, 2016; accepted January 11, 2017; published online March 6, 2017

Abstract Network function virtualization (NFV) is envisioned as one of the critical technologies in 5th-Generation (5G) mobile networks. This paper investigates the virtual network function forwarding graph (VNF-FG) design and virtual network function (VNF) placement for 5G mobile networks. We first propose a two-step method composed of flow designing and flow combining for generating VNF-FGs according to network service requests. For mapping VNFs in the generated VNF-FG to physical resources, we then modify the hybrid NFV environment with introducing more types of physical nodes and mapping modes for the sake of completeness and practicality, and formulate the VNF placement optimization problem for achieving lower bandwidth consumption and lower maximum link utilization simultaneously. To resolve this problem, four genetic algorithms are proposed on the basis of the frameworks of two existing algorithms (multiple objective genetic algorithm and improved non-dominated sorting genetic algorithm). Simulation results show that Greedy-NSGA-II achieves the best performance among our four algorithms. Compared with three non-genetic algorithms (random, backtracking mapping and service chains deployment with affiliation-aware), Greedy-NSGA-II reduces 97.04%, 87.76% and 88.42% of the average total bandwidth consumption, respectively, and achieves only 13.81%, 25.04% and 25.41% of the average maximum link utilization, respectively. Moreover, using our VNF-FG design method and Greedy-NSGA-II together can also reduce the total bandwidth consumption remarkably.

Keywords network function virtualization, VNF-FG design, VNF placement, multi-objective optimization, genetic algorithm

Citation Cao J Y, Zhang Y, An W, et al. VNF-FG design and VNF placement for 5G mobile networks. Sci China Inf Sci, 2017, 60(4): 040302, doi: 10.1007/s11432-016-9031-x

1 Introduction

In traditional mobile networks, operators suffer from many limitations on operating and managing the explosive-increasing services because the deployed equipments in the networks are specialized. They have to deploy a large amount of equipments for meeting various network services. This results in the tremendous costs on equipment deployment and inflexible services providing. Network function virtualization (NFV) [1] is envisioned as one of significant technologies that play important roles in 5th-Generation (5G) mobile networks.

* Corresponding author (email: anwei@iie.ac.cn)

NFV is a network architecture technology that virtualizes the entire classes of network node functions into building blocks that may connect or chain together for creating network services. Its goal is to achieve multiple network functions by using virtualization technologies for decoupling physical equipment and software by the way of replacing hardware-based network functions with software-based virtual network functions (VNFs). A series of VNFs constitute a VNF forwarding graph (VNF-FG) with the help of logical links [2]. VNF-FGs are designed by service providers which receive network service requests and provide specific network services. VNF-FG design has a significant impact on VNF placement and the efficiency of VNF placement affects the performance of NFV seriously [2]. Therefore, service providers aim to find the optimized VNF placement for VNF-FGs based on multiple network service requests and map multiple VNF-FGs to underlying resources.

VNF placement has some close relations with two hot concepts of virtual machine placement (VM-P) and virtual network embedding (VNE). As for VMP, numerous works have been carried out [3–5]. However, these results cannot be applied to VNF placement directly because of the great differences between them: (i) A VNF can also be instantiated by the service ability of a virtual or real network function (NF), which leads to some new mapping relations for VNF placement; (ii) VMP is usually in the scope of a single cloud, while VNF placement is in a much wider scope including multiple clouds as well as physical nodes and links among them. As to VNE, VNF placement can usually be thought as a special kind of the VNE but with some distinctive natures [6]: (a) The VNE only considers resource constraints of physical nodes when performing node mapping, while VNF placement needs to consider resource constraints of virtual machines (VMs) as well as physical nodes that host the VMs; (b) The VNE usually map virtual nodes in the same virtual network to different physical nodes for achieving load balance in physical servers and reducing the risks of physical network faults, while in VNF placement problem it is possible to map VNFs in the same VNF-FG to the same physical node with consideration of the existence of multi-function telecom clouds (MFTCs) [7]. Due to the aforementioned differences, VNF placement has attracted a lot of attentions in recent two years [7–12]. However, these existing works on VNF placement do not consider more types of physical nodes and mapping modes for VNF mapping in NFV environment, or they lack of the theoretical works in the overall optimization.

In this paper, we study the VNF-FG design and VNF placement for 5G mobile networks. The main contributions of this paper are as follows: (i) We propose a two-step method composed of flow designing and flow combining for generating VNF-FGs according to network service requests. (ii) For mapping VNFs in the generated VNF-FG to physical resources, we modify the hybrid NFV environment with introducing more types of physical nodes and mapping modes for the sake of completeness and practicality, and formulate the VNF placement optimization problem for achieving lower bandwidth consumption and lower maximum link utilization simultaneously. To resolve this problem, four genetic algorithms are proposed on the basis of the frameworks of two existing algorithms (multiple objective genetic algorithm and improved non-dominated sorting genetic algorithm). (iii) Extensive simulations show that our algorithm named Greedy-NSGA-II achieves the best performance, which can converge and find the solution quickly, and the total bandwidth consumption can be reduced more using our VNF-FG design method and Greedy-NSGA-II together.

The rest of this paper are as follows. Section 2 introduces the backgrounds and related works. In Section 3, we propose a method for generating VNF-FGs. We modify the hybrid NFV environment and formulate the VNF placement problem in Section 4 and provide our improved genetic algorithms in Section 5. Section 6 shows the simulation results and we conclude this paper in Section 7.

2 Backgrounds and related works

As defined in [2], a network service (NS) is composed of a number of NFs and implemented under the interactions of these NFs. A simple NS can be implemented using point to point links, while more complex one may be achieved by using one or more VNF-FGs. Within the VNF-FG, a set of packets traversing the same path is called a *packet flow.* Service provider receives network service requests from

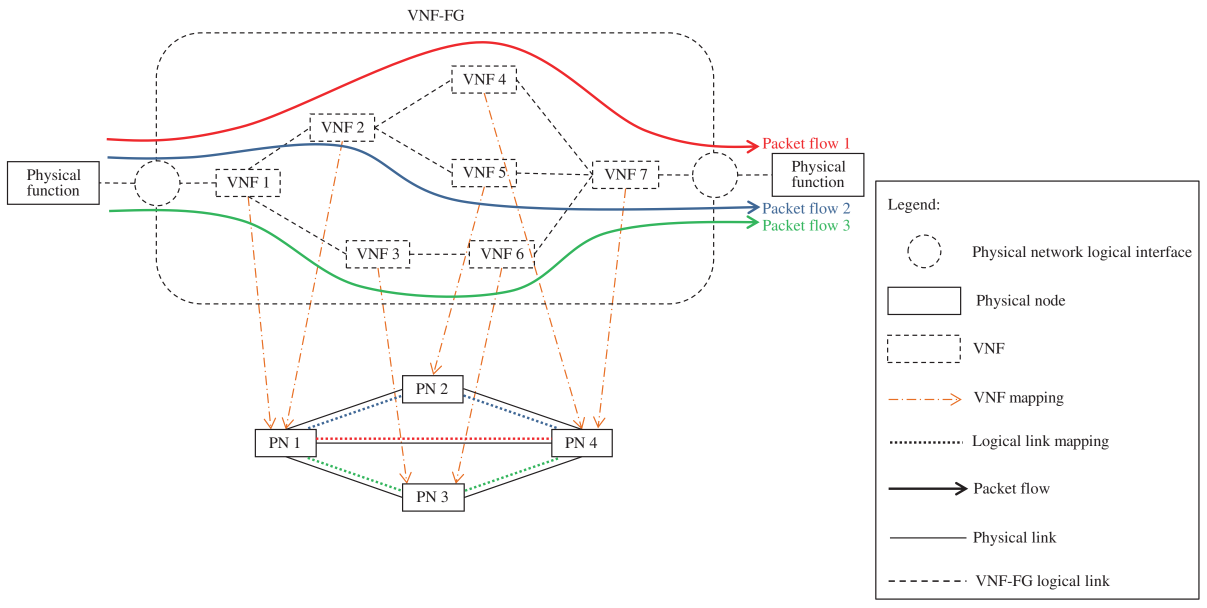


Figure 1 (Color online) The illustration of a VNF-FG and VNF placement.

tenants, design VNF-FGs, and then search the optimized VNF placement.

Many works have been carried out on VNF-FG design and VNF placement. Ref. [9] emphasized that the chaining of VNFs could influence the total data rate, the total number of VNFs and total latency, and this paper also proposed a heuristic method for designing the chaining of VNFs. The main idea is to sort the NFs according to the ratio of the outgoing to the incoming flows and then place them into the chain in turn. Ref. [11] proposed the concept of Cluster-Factor, and a small Cluster-Factor was expected when performing service decomposition. Usually, the goal of VNF placement is to minimize the resource consumption and enhance the reliability. In [8], the concept of hybrid NFV environment is proposed, and a method is presented for obtaining the mapping result within a short time and responding to changes in demand effectively. Nevertheless, the concept of hybrid NFV environment was relatively simple with dividing physical nodes into two categories: service nodes and server nodes, which would severely limit its mapping method when applied in the actual networks. The modeling works of [9] for VNF placement involved multiple optimization objectives, and the authors showed the influence between the remaining data rate, the number of used nodes and the latency of the paths by performing a Pareto set analysis. Although it was effective to make VNF placement decisions, the independence of the objectives might still lead to serious unbalance because of the inherent trade-off between each other. The main contribution of [11] was to optimize the VNF decomposition and mapping jointly with simply summing the objectives together. This could only alleviate but could not fundamentally eliminate the unbalance. Ref. [7] studied how to jointly optimize the VNF-FG design and VNF placement such that the TBC could be minimized. Specially, the concept of VNF combination was first proposed in [7]. VNF combination implies that different VNFs can be mapped to the same MFTC. Intuitively, VNF combination can reduce the bandwidth consumption between the combined VNFs without causing extra overhead; however, it is not necessarily guaranteed to reduce the TBC with an instance that can be found in [7]. Unfortunately, their work lacks of the modeling work with consideration of VNF combination.

The above imperfections motivate us to propose some novel works for VNF-FG design and VNF placement the detail of which will be shown in the following sections.

Figure 1 presents an instance of a VNF-FG, packet flows and VNF placement. In this figure, the service provider designs a VNF-FG implementing an end-to-end NS between two physical network functions. The VNF-FG is composed of seven VNFs and eight logical links, and three packet flows traverse through the VNF-FG as the output. During the process of VNF placement, these seven VNFs are mapped to four physical nodes and the eight logical links are mapped to five physical links.

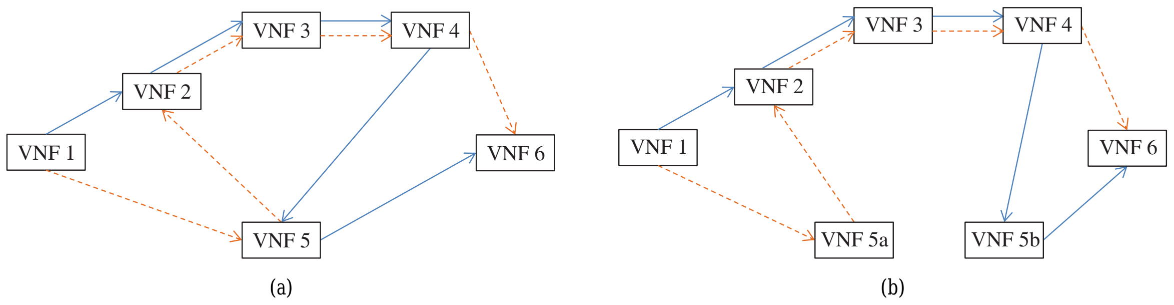


Figure 2 (Color online) The illustration of flow combining. (a) The union of flows; (b) eliminating circles.

3 VNF-FG design

To place VNF-FGs on physical networks, we should focus on the features:

- The same network service can be implemented with multiple methods.
- A VNF can be mapped to different platforms.
- Implementation can be optimized for different objectives.
- There is no need to consider the details of VNF implementation.
- The load balance in infrastructure components should not be ignored.
- Advanced resource optimization can be implemented by jointly optimizing VNF-FG design and VNF placement.

Before VNF-FG design, it is assumed that the number of flows in a VNF-FG, VNF instances needed in each flow and the ratio of outgoing and incoming flows of each VNF instance are known. Some constraints in relative orders between different VNF instances may exist, in other words, a flow should pass through a VNF instance earlier than the other. Each VNF instance is with a limited ability of processing requests. Two steps are presented for VNF-FG design: (1) flow designing, and (2) flow combining.

Step 1. It is assumed that m VNF instances exist in a flow. Construct vector $(x_1, x_2, \ldots, x_m)$, where x_i represents the order of ith VNF instance. Because of the constraints in terms of the relative orders of VNF instances, we present them in the form of inequalities. For example, if a flow should pass through ith VNF instance earlier than jth VNF instance, we have $x_i < x_j$. Motivated by [9], the goal of this step is to minimize the sum of the product of the order of each VNF instance and the ratio of outgoing and incoming flows of the VNF instance as shown in Formula (1):

$$\min \sum_{v \in f} \mathrm{order}(v) \times \frac{\mathrm{outgoing}(v)}{\mathrm{incoming}(v)}, \tag{1}$$

where v, order(v), outgoing(v) and incoming(v) represent a VNF on flow f, the order of v in f, the rate of outgoing flows from v and the rate of incoming flows to v, respectively. Thus, it is an Integer Linear Programming model. We can solve it by using existing tools, for example, *lingo*. If a flow goes through VNF 4, VNF 2, VNF 3 and VNF 1 in turn, we gain a vector $(4, 2, 3, 1)$ which is the final representation of the flow.

There exist three basic topological components of VNF-FGs: line, bifurcated path with different endpoints and bifurcated path with a single endpoint [13]. Components consisting of a single function or start/end point of flows are stored as one node of the VNF-FG. Here, we generate a VNF-FG without consideration of those VNFs which can be placed at any place in a flow.

Step 2. Firstly, we calculate the union of flows, which is a graph union operation actually. Figure 2(a) is the union of flow $f1 = (1, 2, 3, 4, 5, 6)$ and flow $f2 = (1, 5, 2, 3, 4, 6)$. In Figure 2, the blue arrows represent $f1$ and the orange arrows indicate $f2$. Considering that VNF-FG is a directed graph, secondly, judge whether circles exist in the united graph using the topological sort algorithm. If the graph can be sorted, there is no circle in it. Otherwise, circles need to be eliminated. To eliminate the circles, pick up the flows on these circles. As shown in Figure 2(a), VNF 2, VNF 3, VNF 4 and VNF 5 construct a circle. These four VNFs belong to both $f1$ and $f2$, and $f1$ and $f2$ are both on the circle. For these flows, there might be conflicts in the relative orders of different VNF instances, and we call these VNF instances *key instances*. In Figure 2(a), VNF 5 is the key instance because VNF 5 is after VNF 2, VNF 3 and VNF 4 in $f1$ while it is before them in $f2$. We decompose each key instance into k instances, where k represents the number of flows which are on a circle and pass through the key instance. For VNF 5 in Figure 2(a), $k = 2$ because both $f1$ and $f2$ are on the circle and pass through VNF 5. As a result, VNF 5 is decomposed to VNF 5a and VNF 5b. Subsequently, the links should be decomposed as well according to the affiliations. Finally, the result of eliminating circles from Figure 2(a) is presented in Figure 2(b), where VNF 5a only connects the links on $f2$ and VNF 5b merely connects the links of $f1$.

4 Formulating VNF placement in hybrid NFV environment

In this section, we first modify the concept of hybrid NFV environment with introducing more types of physical nodes and mapping modes, and then propose a multi-constraint and multi-objective optimization formulation for VNF placement in the hybrid NFV environment.

4.1 The hybrid NFV environment

In the transition from traditional networks to NFV networks, it is necessary for functions running on virtual resources to co-exist with those on physical resources [14]. Section 2 has indicated that the concept of hybrid NFV environment defined in [8] is relatively simple because of ignoring that the partition of physical nodes can be more detailed. Actually, the requests and mapping rules in terms of different physical nodes should be different. Here we provide a more comprehensive definition of hybrid NFV environment, which includes three aspects: (a) the hybrid of three kinds of physical nodes, (b) the hybrid of two kinds of virtual requests, and (c) the hybrid of three kinds of mapping modes.

(a) The hybrid of three kinds of physical nodes. Three kinds of physical nodes refer to: (1) The physical nodes which can directly provide service capabilities, but VMs cannot be created on, namely SNs (service nodes); (2) The physical nodes which cannot directly provide service capabilities, but VMs providing the same function can be created on, namely SFTCs (single-function telecom clouds); (3) The physical nodes which cannot directly provide service capabilities, but VMs providing different functions can be created on, namely MFTCs. In other words, an SFTC can only provide a kind of NF, while an MFTC can provide several kinds of NFs. The division between SFTC and MFTC is meaningful, because in some future concept such as MEC (mobile edge computing) or Cloudlets, it is possible to deploy small TCs that are implemented with a single NF at the edge of the network.

(b) The hybrid of two kinds of virtual requests. A VNF can be instantiated with either a VM or the service ability of an NF via VM requests or service requests. Two kinds of virtual requests (namely VNF requests) refer to VM requests and service requests. A VM request is for an instance of a specific VM, while a service request is for a sort of specific service capability, no matter from a VM or an SN.

(c) The hybrid of three kinds of mapping modes. Three kinds of mapping modes refer to: (1) A VM request can be directly mapped to an SFTC or an MFTC which can create the VM; (2) A service request can be mapped directly to a physical node that is capable of providing the corresponding service capability, which is called direct service request mapping; (3) A service request can be mapped to a VM that can provide the service capability, and then the VM is mapped to an SFTC or an MFTC that can create such a VM, which is called indirect service request mapping.

The mapping modes in the hybrid NFV environment are depicted in Figure 3. Compared with the previous works [7, 8], the novel mapping modes make the formulating work more complex.

4.2 Problem formulation

In this subsection, we first give out some assumptions which are reasonable for simplifying some conditions in our formulation: (1) All VNF-FGs are known in advance and processed in order, which means it is a static problem. (2) A service request has the priority to be mapped directly. If it cannot be achieved, indirect mapping of the service request could be performed. That is to say, we should make full use of

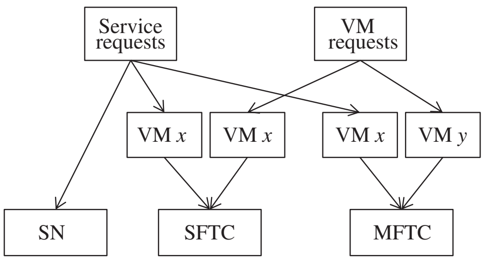


Figure 3 The mapping modes in the hybrid NFV environment.

SNs. (3) A VM can only provide one kind of service capability. Besides the above, we also ignore the background traffic, which is similar with previous works [8, 9, 11].

In the process of mapping virtual networks to physical network, we set two goals which are to minimize the total bandwidth consumption and the maximum link utilization, respectively. The first goal is to save the bandwidth resources, and the second goal aims to achieve load balance. The two goals are in conflict to some extent. For two given physical nodes, there may be multiple paths between them. In order to pursue the first goal, we tend to choose the shortest path. But when the request is in a large scale, it may cause that some links are overload and some links are idle, which is unreasonable. To avoid this, the second goal is proposed. Formulas (2) and (3) show the two optimization goals, respectively:

$$\min \textstyle\sum_{l_S \in E_S} u(l_S), \tag{2}$$

$$\min \max_{\forall l_S \in E_S} \tfrac{u(l_S)}{D(l_S)}, \tag{3}$$

where E_S, $u(l_S)$ and $D(l_S)$ represent the set of physical links, the bandwidth that physical link l_S has allocated and the bandwidth that physical link l_S can provide, respectively.

Constraints of VNF placement in hybrid NFV environments are divided into four categories, namely, link capacity constraints, node resource constraints, mapping relation constraints and flow constraints, which are introduced in details in the following.

4.2.1 *Link capacity constraints*

$$\forall l_S \in E_S: \ u(l_S) = \textstyle\sum_{i=1}^{r} \sum_{f_l \in F_i} \sum_{l_V \in f_l} \sum_{p \in P_M(l_V)} x^p(l_V) y^p(l_S) b(l_V), \tag{4}$$

$$\forall l_S \in E_S: \ u(l_S) \leqslant D(l_S). \tag{5}$$

r is the number of VNF-FGs, F_i represents the set of flows in VNF-FG R_i, $P_M(l_V)$ denotes the set of alternative physical paths where virtual link l_V is mapped, and $b(l_V)$ is the bandwidth demand of virtual link l_V. $x^p(l_V)$ and $y^p(l_S)$ are both binary variables. If virtual link l_V is mapped to physical path p, $x^p(l_V) = 1$; otherwise, $x^p(l_V) = 0$. If physical link l_S is along physical path p, $y^p(l_S) = 1$; otherwise, $y^p(l_S) = 0$.

A physical link can be allocated to multiple virtual links, so Eq. (4) shows the allocated bandwidth on physical link l_S. Inequation (5) indicates that the allocated bandwidth on l_S should not exceed the total available bandwidth on l_S.

4.2.2 *Node resource constraints*

Three kinds of physical nodes exist in the hybrid NFV environment. Consequently, different types of physical nodes have different node resource constraints.

$$\textstyle\sum_{f \in F} (v(n_S, f) \times c_\gamma^{\mathrm{VM}}(f)) \leqslant C_\gamma(n_S), \ \forall n_S \in V_S(M), \tag{6}$$

$$v(n_S, f) = d(n_S, f) + n(n_S, f), \tag{7}$$

$$d(n_S, f) = \textstyle\sum_{i=1}^{r} \sum_{n_V \in V_i} m_1(n_V, n_S), \tag{8}$$

$$n(n_S, f) = \textstyle\sum_{i=1}^{r} \sum_{n_V \in V_i} m_2(n_V, n_S). \tag{9}$$

F is the set of all network functions. $v(n_S, f)$, $d(n_S, f)$ and $n(n_S, f)$ are the number of the VMs created on physical node n_S for providing network function f, the number of the VMs created on physical node n_S for providing network function f and for indirect service request mapping, and the number of the VMs created on physical node n_S for providing network function f and for direct VM request mapping, respectively. $c_\gamma^{\mathrm{VM}}(f)$ and $C_\gamma(n_S)$ denote the amount of physical resource type γ needed when creating a VM that can provide network function f, and the amount of physical resource type γ that physical node n_S can provide, respectively. $V_S(M)$ represents the set of MFTCs. V_i is the set of virtual nodes in G_i which is the virtual network topology that VNF-FG R_i corresponds to. $m_1(n_V, n_S)$ and $m_2(n_V, n_S)$ are both binary variables. If service request n_V is indirectly mapped to an SFTC or MFTC n_S, $m_1(n_V, n_S) = 1$; otherwise, $m_1(n_V, n_S) = 0$. If VM request n_V is mapped to an SFTC or MFTC n_S, $m_2(n_V, n_S) = 1$; otherwise, $m_2(n_V, n_S) = 0$.

Inequation (6) states the node resource constraint for MFTCs which implies that the total amount of physical resources γ allocated for creating various VMs on an MFTC n_S should not exceed the total amount of physical resources γ that n_S can provide. Here, we assume that only one service request can be mapped to a given VM. The calculation of $v(n_S, f)$ is shown in (7)–(9).

$$\forall n_S \in V_S(S),\ \forall f \in F:\ v(n_S, f) \times c_\gamma^{\mathrm{VM}}(f) \leqslant C_\gamma^f(n_S), \tag{10}$$

where $V_S(S)$ is the set of SFTCs and $C_\gamma^f(n_S)$ represents the amount of physical resource type γ that physical node n_S can provide to create the VM providing network function f.

Inequation (10) is the node resource constraint for SFTCs. Inequation (10) indicates that the total amount of physical resources γ allocated for creating specific VMs on an SFTC n_S should not exceed the total amount of physical resources γ that n_S can provide. Similar to an MFTC, VMs created on n_S for the specific network function f are divided into two parts. Note that, if VMs providing network function f cannot be created on n_S (n_S is an SFTC and can only create VMs providing a specific network function), $C_\gamma^f(n_S) = 0$. This constraint avoids meaningless mapping results. The calculation of $v(n_S, f)$ is also according to (7)–(9).

$$\forall n_S \in V_S(P):\ \sum_{i=1}^{r} \sum_{n_V \in V_i} c_\beta^S(n_V) m_3(n_V, n_S) \leqslant D_\beta(n_S). \tag{11}$$

$V_S(P)$ is the set of SNs. $c_\beta^S(n_V)$ and $D_\beta(n_S)$ represent the capability of service type β needed when service request n_V is directly mapped to a physical node, and the capability of service type β that an SN n_S can provide, respectively. $m_3(n_V, n_S)$ denotes a binary variable. If service request n_V is directly mapped to an SN n_S, $m_3(n_V, n_S) = 1$; otherwise, $m_3(n_V, n_S) = 0$.

Inequation (11) is the resource constraint for physical nodes that can only be used for direct service request mapping. Inequation (11) implies that the total capability of service type β allocated by physical node n_S, which can only be used for direct service request mapping, should not exceed the total capability of service type β that n_S can provide. Similar with Inequation (10), this constraint also avoids meaningless mapping results.

4.2.3 *Mapping relation constraints*

$$\forall l_V \in E_i:\ \textstyle\sum_{p \in P_M(l_V)} x^p(l_V) = 1, \tag{12}$$

$$\forall n_V \in V_i,\ i = 1, 2, 3, \ldots, r:\ \textstyle\sum_{n \in V_S} m(n_V, n) = 1, \tag{13}$$

$$v_{ix}, v_{iy} \in V_i, i = 1, 2, 3, \ldots, r: \textstyle\sum_{n \in V_S(M)} m(v_{ix}, n) m(v_{iy}, n) = 0. \tag{14}$$

E_i is the set of virtual links in the virtual network topology G_i. V_S represents the set of physical nodes, and $V_S = \{V_S(P), V_S(S), V_S(M)\}$. $m(n_V, n_S)$ denotes a binary variable. If a request n_V (a VM request or a service request) is mapped to the physical node n_S, $m(n_V, n_S) = 1$; otherwise, $m(n_V, n_S) = 0$.

Eq. (12) indicates that each virtual link can only be mapped to one physical path. Eq. (13) implies that each virtual request (a VM request or a service request) can only be mapped to one physical node.

Because of taking VNF combination into account, we do not adopt the constraint which ensures that virtual requests (VM requests or service requests) in the same VNF-FG cannot be mapped into the same physical node. Moreover, as mentioned in [7], VNF combination is not necessarily guaranteed to reduce the TBC. So we use Formulas (2) and (3) to ensure that the mapping result is the best no matter VNF combination is performed or not. However, mapping VNFs in the same VNF-FG into the same physical node is not always permitted in practice. For example, the backup nodes in a hot-standby system cannot be mapped to the same physical node. So we adopt (14), which is a precondition and known when the VNF-FG is given. Eq. (14) is an important constraint for VNF combination, which implies that two VNFs in the same VNF-FG may not be mapped to the same physical node.

4.2.4 *Flow constraints*

$$\begin{aligned}&\forall (v_1, v_2) \in F_i, i = 1, 2, \ldots, r;\ \forall n \in V_S : \\ &\qquad m(v_2, n) + \sum_{e \in E_n^{\mathrm{out}}} F(e, v_1, v_2) = m(v_1, n) + \sum_{e \in E_n^{\mathrm{in}}} F(e, v_1, v_2),\end{aligned} \tag{15}$$

$$\forall (v_1, v_2) \in F_i, i = 1, 2, \ldots, r;\ \forall n \in V_S :\ m(v_2, n) + \sum_{e \in E_n^{\mathrm{out}}} F(e, v_1, v_2) \leqslant 1, \tag{16}$$

and

$$m(v_1, n) + \sum_{e \in E_n^{\mathrm{in}}} F(e, v_1, v_2) \leqslant 1. \tag{17}$$

E_n^{out} and E_n^{in} represent the set of physical links that are outgoing from physical node n and the set of physical links that are incoming in physical node n, respectively. $F(e, v_1, v_2)$ is a binary variable. If physical link e is along the path that flow (v_1, v_2) (v_1 and v_2 are the source node and the destination node, respectively) is mapped to, $F(e, v_1, v_2) = 1$; otherwise, $F(e, v_1, v_2) = 0$.

For any flow, except the source and the destination, all nodes that the flow goes through should satisfy the flow conservation equation (15). Inequations (16) and (17) indicate that the mapping result of a flow should be acyclic.

Formula (2) to (17) form a formulation for optimal VNF placement in the hybrid NFV environment, and it is a complicated multi-objective optimization problem. Obviously, this is an NP-hard problem. In the next section, we propose four genetic algorithms as solutions for this problem.

5 Solutions

Our four genetic algorithms are based on multiple objective genetic algorithm (MOGA) [15] and improved non-dominated sorting genetic algorithm (NSGA-II) [16], which are two representative multi-objective optimization genetic algorithms. The frameworks of MOGA and NSGA-II are shown in Figure 4.

Without using the original MOGA and NSGA-II algorithms themselves, we use their frameworks and make some novel modifications. In the frameworks of MOGA and NSGA-II, our works include:

- Encoding: Encoding is a phase in genetic algorithms. Common encoding methods are binary encoding, floating encoding and symbolic encoding. Here, we adopt binary encoding. The node mapping is encoded to a binary matrix. If there are m virtual nodes and n physical nodes, the node mapping matrix is an $m \times n$ matrix. And if the ith virtual node is mapped to the jth physical node, the entry (i, j) of this matrix is 1 and other elements in the ith row are 0. The mapping result of a given virtual link is encoded to a binary matrix. If the physical link between the ith physical node and the jth physical node is along the physical path corresponding to the given virtual link, the entry (i, j) of the matrix is 1.

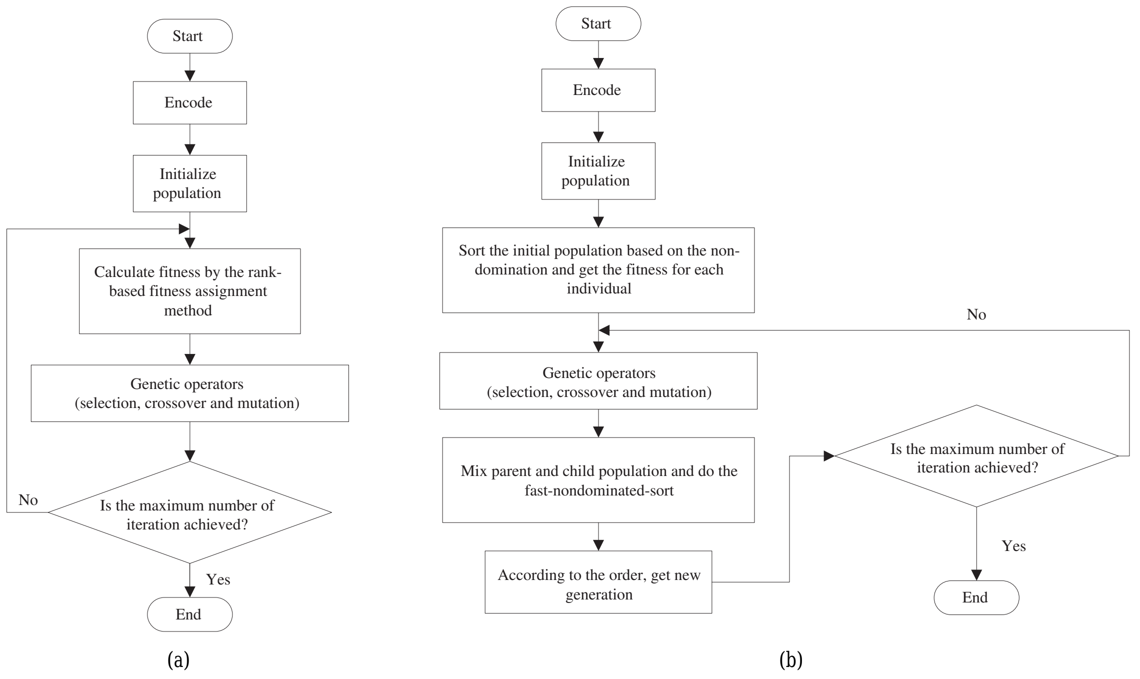

Start
Encode
Initialize population
Calculate fitness by the rank-based fitness assignment method
Genetic operators (selection, crossover and mutation)
Is the maximum number of iteration achieved?
No
Yes
End
(a)
Start
Encode
Initialize population
Sort the initial population based on the non-domination and get the fitness for each individual
Genetic operators (selection, crossover and mutation)
Mix parent and child population and do the fast-nondominated-sort
According to the order, get new generation
Is the maximum number of iteration achieved?
No
Yes
End
(b)

Figure 4 The frameworks of former algorithms. (a) MOGA; (b) NSGA-II.

• Initializing population: Initializing population is a phase in genetic algorithms. Usually, the initial population is produced by random strategy. But in our algorithms, two strategies are applied in the phase: random and greedy. The greedy strategy means that in node mapping, nodes with more residual resources have higher priority; while in link mapping, paths with shorter length have higher priority. No matter which strategy is used in initialization phase, node mapping is processed before link mapping.

Besides the above, we also propose the following works which are different from original MOGA and NSGA-II:

• Checking for feasibility: Crossover and mutation may cause the mapping results that cannot satisfy the constraints in Section 4, so the phase for checking for the feasibility of mapping results is added after crossover and mutation. If there are invalid mapping results, we should correct them so that they can satisfy all the constraints.

• Selecting niche radius: In MOGA, the selection of niche radius is an important process when calculating the fitness, and it directly affects the overall performance of the algorithm. For our problem, the computational complexity of the method to select niche radius in MOGA is relatively high, so the average of Euclidean distances between any two individuals in a population is used as the niche radius. Formula (18) presents the niche radius:

$$\frac{\sum_{x\in P}\sum_{y\in\{P/x\}}\left[(x_1-y_1)^2+(x_2-y_2)^2\right]}{A_n^2}. \tag{18}$$

P is a population with $n=|P|$. x_i and y_i $(i=1,2)$ represent the ith objective of individuals x and y, respectively. A_n^2 indicates the number of all the possible permutations of 2 individuals from P.

Meanwhile, we adopt common rules for crossover and mutation in our genetic algorithms. In the process of crossover, the one-point crossover is performed based on the crossover probability. A pair of individuals are selected randomly and only one crossover point is set randomly. And then the bits after the crossover point in one individual are exchanged with these in another individual. In the process of mutation, the mutation is performed based on the mutation probability. For one individual to be mutated, its mutation point is set randomly. We flip the bit in the mutation point with values changing from 1 to 0 or from 0 to 1.

Finally, based on the frameworks of MOGA and NSGA-II and our novel works, four algorithms are

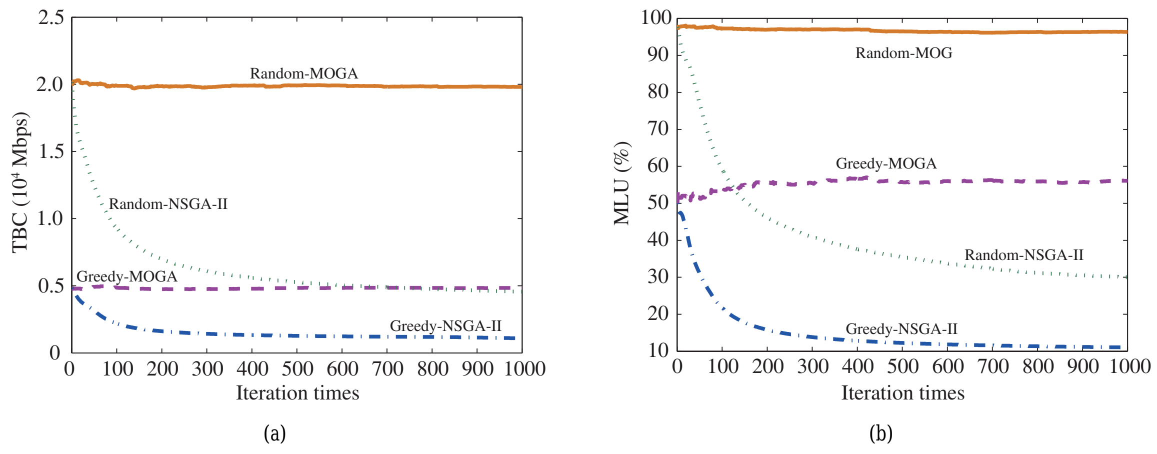


Figure 5 (Color online) The relations between the objectives and the iteration times for our 4 algorithms. (a) TBC vs. iteration times; (b) MLU vs. iteration times.

proposed. These algorithms are named based on the strategy used to initialize population and the whole algorithm framework. The algorithm which based on the framework of MOGA and uses random strategy to get the initial population is named Random-MOGA. Similarly, other three algorithms are named Greedy-MOGA, Random-NSGA-II and Greedy-NSGA-II, respectively.

6 Evaluation

In this section, we carry out extensive simulations to evaluate the performance of the proposed algorithms. Our simulation tools are developed by using VC++ 6.0.

6.1 Simulation settings

The physical network topology used in the simulation is the NSFNet, which is composed of 14 nodes and 21 links and often used in simulations for network researches. In our simulations, we set 2 SNs, 2 SFTCs and 10 MFTCs. We refer to five kinds of service capabilities and three kinds of network functions in total, and we assume that MFTCs can create various VMs for providing the five kinds of service capabilities and the three kinds of network functions. For the four genetic algorithms, the number of individuals in each generation is set to 30, and the iteration times is set to 1000. Two kinds of node resources are considered, including CPU and memory. Two metrics including TBC and MLU are adopted.

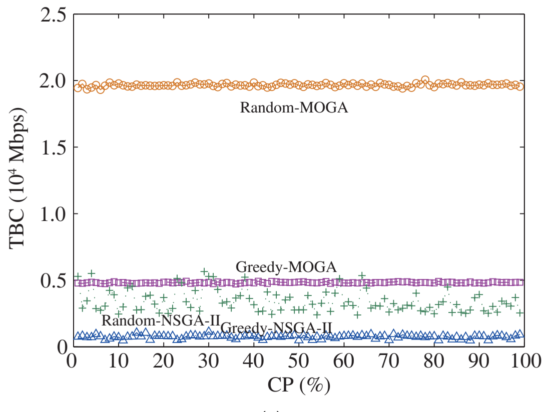
6.2 Results analysis

6.2.1 *Comparisons of the proposed four algorithms*

In the subsection, we carry out 3 groups of simulations, where 100 VNF-FGs are fixed and input before mapping. In the physical network, 3 links are set to 1 Gbps and other 18 links are set to 5 Gbps.

(1) The impact of iteration times. We first study the impact of the iteration times on the performance of the four genetic algorithms. Both the crossover probability (CP) and the mutation probability (MP) are set to 30%. Simulation results are shown in Figure 5.

From Figure 5(a) we can see that, as the iteration times increases, the TBCs of MOGA-based algorithms change slightly, which are always around 20000 Mbps and 5000 Mbps, respectively. That means the iterations in MOGA-based algorithms nearly make no sense to decrease the TBC. On the contrary, we can observe that the increase of the iterations in NSGA-II-based algorithms can optimize the objective effectively. Importantly, when the iteration times is approaching 1000, the TBCs of the NSGA-II-based algorithms get stable. On the whole, Greedy-NSGA-II achieves the best TBC performance among all the four algorithms, and for the same framework (MOGA or NSGA-II) the greedy strategy is better than the random strategy.



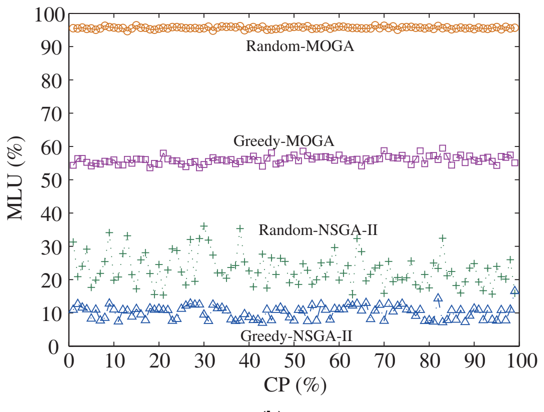


Figure 6 (Color online) The relations between the objectives and the CP for our 4 algorithms. (a) TBC vs. CP; (b) MLU vs. CP.

Results and observations from Figure 5(b) are similar with those from Figure 5(a). Specially, when the iteration times is over 100, the MLU performance of Random-NSGA-II can be even better than that of Greedy-MOGA. This is resulted from the advantage of the NSGA-II framework over the MOGA framework.

In Figure 5, we can also observe that the starts of the curves of greedy-related algorithms are below those of random-related algorithms. This is because the greedy strategy applied in population initialization can produce the initial population with smaller objectives than the random strategy.

In NSGA-II-based algorithms, the individuals are ranked and sorted before the selection phase, so the individuals with lower TBC as well as lower MLU have the higher priority to be chosen as parent individuals. Thanks to this, good features can be transferred to next generation. This is the reason that NSGA-II-based algorithms have better performance than MOGA-based ones with the same initialization strategy (greedy or random).

(2) The impact of the crossover probability. We then study the impact of the CP on the performance of the four genetic algorithms. The MP is fixed to 30%, while the CP is varied from 1% to 99%. Simulation results are shown in Figure 6. From Figure 6, it can be observed that, as the CP increases, there is no clear upward or downward trend about the TBC and MLU for the 4 algorithms. That means the CP makes little sense to decrease or increase the performance of the algorithms. Other results are similar with those in Figure 5, including that Greedy-NSGA-II always achieves the best performance. It is also observable that the performance jitter of the NSGA-II-based algorithms is much more obvious than that of the MOGA-based algorithms, which indicates that the former ones are more sensitive to the CP than the later ones.

(3) The impact of the mutation probability. We also study the impact of the MP on the performance of the four genetic algorithms. The CP is fixed to 30%, while the MP is varied from 1% to 99%. Simulation results are shown in Figure 7. Figure 7 shows that, there is a little upward trend about the TBC and MLU for the MOGA-based algorithms when the MP is small, and then the curves are stable. While for the NSGA-II-based algorithms, as the MP increases over 30%, the upward trend about the TBC and MLU become obvious. On the whole, Greedy-NSGA-II achieves the best TBC and MLU performance among all the four algorithms. And observations about the performance jitter are similar with those in Figure 6.

From the above we can conclude that Greedy-NSGA-II is the best one in the four proposed algorithms. Its iteration times can be set to 1000, and its MP should not be higher than 30%.

6.2.2 *Comparisons of Random, BM, SCDWA and Greedy-NSGA-II*

To prove the effectiveness of Greedy-NSGA-II, we compare its TBC and MLU performance with three algorithms: random, backtracking mapping [11] (BM) and service chains deployment with affiliation-

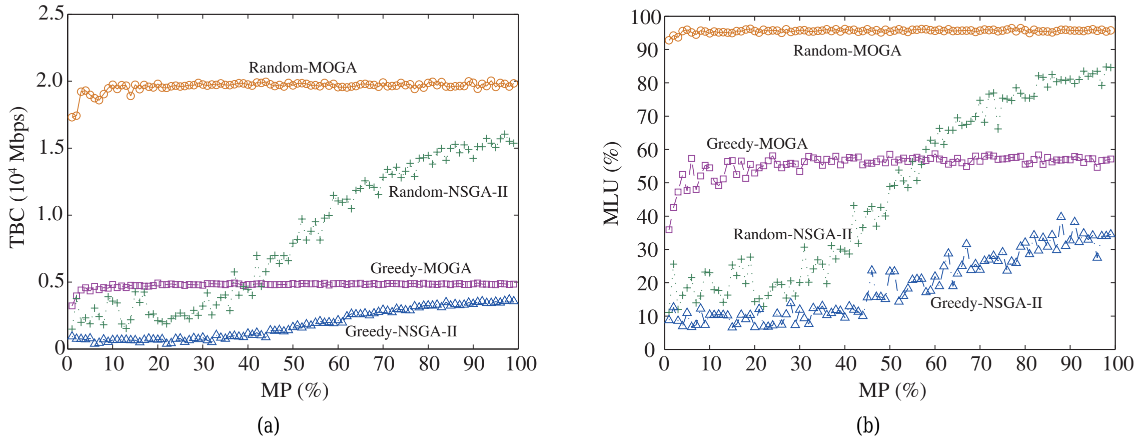


Figure 7 (Color online) The relations between the objectives and the MP for our 4 algorithms. (a) TBC vs. MP; (b) MLU vs. MP.

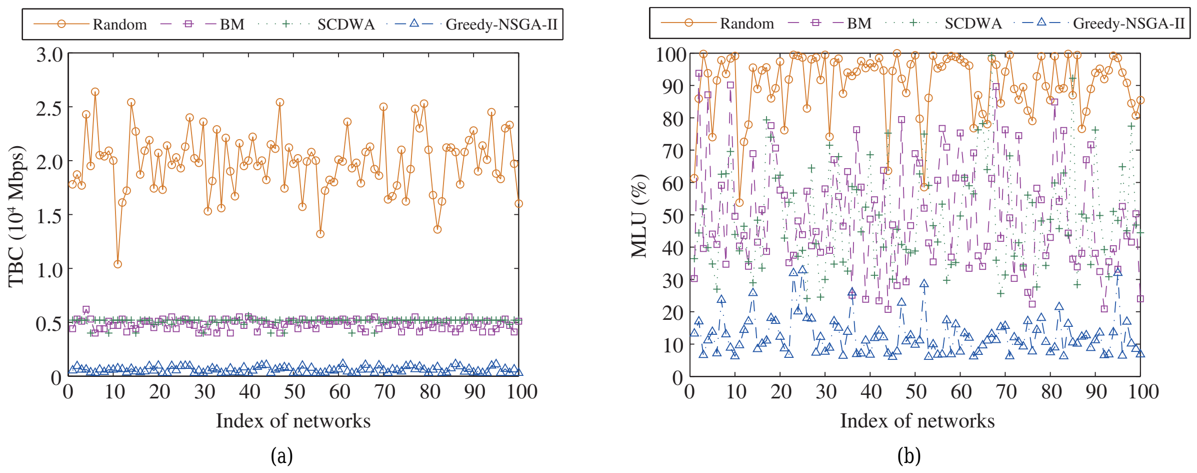
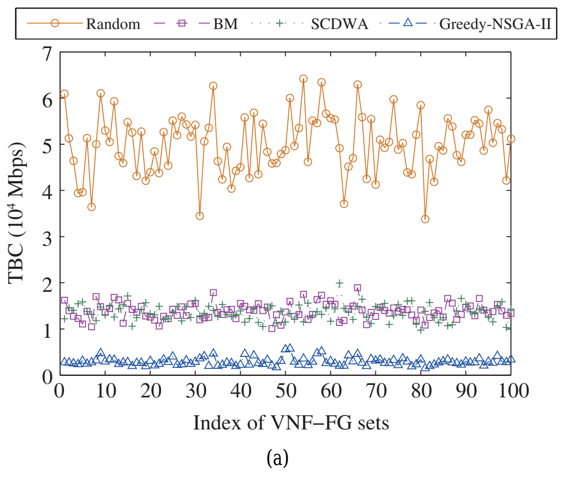


Figure 8 (Color online) The relations between the objectives and physical networks for Random, BM, SCDWA and Greedy-NSGA-II. (a) TBC vs. the index of networks; (b) MLU vs. the index of networks.

aware [17] (SCDWA). As the term suggests, Random adopts a random strategy both in node mapping and link mapping. The authors of [11] propose an algorithm named decomposition selection-backtracking mapping (DSBM). Decomposition Selection and Backtracking Mapping are independent each other and Decomposition Selection is irrelevant to this paper (it is about VNF-FG decomposition, which is another topic), so we pick out Backtracking Mapping as a contrastive VNF placement algorithm. In [17], the main idea of SCDWA is to place VNFs considering their affiliation in requested service chains. According to Figures 6 and 7, it can be observed that the CP makes little sense to decrease or increase the performance of Greedy-NSGA-II, and it is observable that when the MP is 30%, the performance of Greedy-NSGA-II is the worst effect we can accept. So the CP and MP are both set to 30% in Greedy-NSGA-II, and it is fair to compare it with other algorithms.

We carry out two groups of simulations. In the first group, 100 VNF-FGs are definite, while the bandwidths of the physical links in NSFNet are generated randomly for 100 times (thus generating 100 different networks). All the bandwidths are between 1 Gbps and 5 Gbps. For each network, the four algorithms are performed and compared. Simulation results are shown in Figure 8. It is obvious that Greedy-NSGA-II achieves the lowest TBC and MLU among the four algorithms. Specifically, the average TBC value of Greedy-NSGA-II is 587.77 Mbps, only 2.96%, 12.24% and 11.58% of that of Random, BM and SCDWA, respectively. And the average MLU value of Greedy-NSGA-II is 12.49%, only 13.81%, 25.04% and 25.41% of that of Random, BM and SCDWA, respectively.

In the second group, the physical network is the same with that in the last subsection, while 100



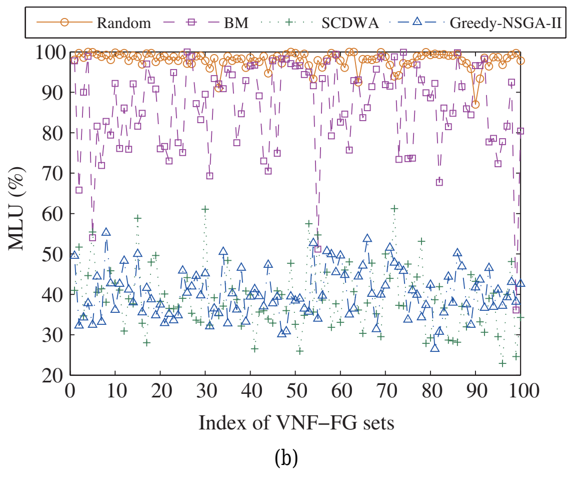


Figure 9 (Color online) The relations between the objectives and VNF-FG sets for Random, BM, SCDWA and Greedy-NSGA-II. (a) TBC vs. the index of VNF-FG sets; (b) MLU vs. the index of VNF-FG sets.

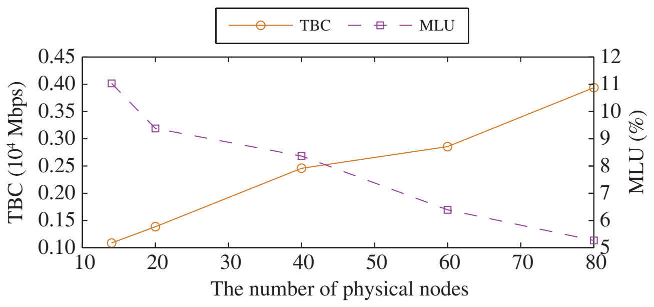


Figure 10 (Color online) The relations between the objectives and the number of physical nodes for Greedy-NSGA-II.

sets of VNF-FGs are generated randomly and each set is composed of 100 VNF-FGs. For each set of VNF-FGs, the four algorithms are performed and compared. Simulation results are shown in Figure 9, where Greedy-NSGA-II still achieves the lowest TBC and MLU among the four algorithms. Specifically, the average TBC value of Greedy-NSGA-II is 2935.52 Mbps, only 5.86%, 21.16% and 21.60% of that of Random, BM and SCDWA, respectively. And the average MLU value of Greedy-NSGA-II is 40.23%, only 41.07% and 46.67% of that of BM and Random, respectively. Although the average MLU value of SCDWA is 39.06% which is slightly lower than that of Greedy-NSGA-II, the remarkable advantage of Greedy-NSGA-II in terms of TBC dwarfs this slight gap to be negligible.

In sum, we can conclude that, Greedy-NSGA-II is much more effective than Random, BM and SCDWA to reduce the TBC and MLU, with different networks and sets of VNF-FGs.

To evaluate the effectiveness of Greedy-NSGA-II further, we raise the number of physical nodes for evaluations based on the simulations above. The results shown in Figure 10 indicate that the TBC increases and the MLU deceases with the augment of physical nodes. With the expansion of the physical network, physical paths that logical links are mapped to are likely to become longer, which leads to the increase of TBC. At the same time, the expansion of the physical network makes the traffic distribute to more physical links, resulting in the decrease of MLU.

It has been proved that the computational complexity of NSGA-II is $O(mN^2)$ [16], where m is the number of objectives and N is the number of individuals in a population. For our Greedy-NSGA-II, the number of objectives is 2 and the number of individuals in a population is 30, so its computational complexity is acceptable. Actually, all the simulations are completed on a computer with 2 Intel Xeon E5-2609 CPUs running at 2.50 GHz, and each group of simulations spends about 10 min. It can be observed in Figure 5 that after the iteration time reaches around 200, the curves of Greedy-NSGA-II change slowly. This implies that the algorithm can stop at around 200th round for remarkably saving

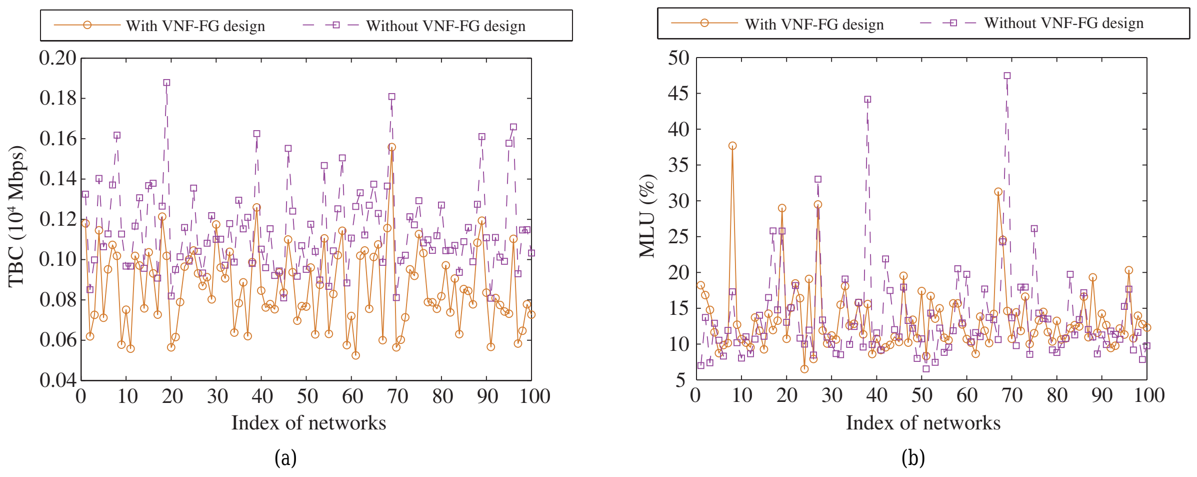


Figure 11 (Color online) The comparisons between the objectives with VNF-FG design and without VNF-FG design. (a) TBC with VNF-FG design vs. TBC without VNF-FG design; (b) MLU with VNF-FG design vs. MLU without VNF-FG design.

the consumed time when the results meet our requirements in accuracy.

6.2.3 *The necessity of the VNF-FG design*

To prove the necessity of the VNF-FG design and the effectiveness of our method, we carry out a group of comparison simulations to compare the TBC and MLU performance of the placements with and without the proposed VNF-FG design method in Section 3. In the placement with VNF-FG design, the number of flows in a VNF-FG, VNF instances needed in each flow and the ratio of outgoing and incoming flows of each VNF instance are known. We perform VNF-FG design using the method mentioned above and VNF-FG placement with using Greedy-NSGA-II where the CP and the MP are both set to 30%. In the placement without VNF-FG design, the VNF-FG is generated randomly and the placement algorithm is Greedy-NSGA-II where the CP and the MP are both set to 30% as well. In the simulations, the bandwidths of the physical links in NSFNet are generated randomly for 100 times (thus generating 100 different networks), and all the bandwidths are between 1 Gbps and 5 Gbps. For each network, we run Greedy-NSGA-II for the VNF-FGs with and without VNF-FG design. Simulation results are shown in Figure 11. It can be observed that the placement with the VNF-FG design achieves lower TBC and MLU. Specifically, the average TBC value of the placement with VNF-FG design is 865.22 Mbps, only 75.46% that of without VNF-FG design. However, the average MLU value of the placement with VNF-FG design is 13.48%, similar with that of without VNF-FG design.

7 Conclusion

In this paper, we studied the VNF-FG design and VNF placement for 5G mobile networks. We have proposed a two-step method consisting of flow designing and flow combining to generate VNF-FGs according to network service requests. The concept of hybrid NFV environment has been modified with more types of physical nodes and mapping modes for the sake of completeness and practicality. On the basis of this concept, we have formulated the VNF placement as an optimization problem with consideration of VNF combination for reducing the total bandwidth consumption and the maximum link utilization. To resolve this problem, four genetic algorithms have been presented by using the frameworks of two existing MOGA and NSGA-II. Extensive experiments have been carried out to illustrate the performance of our proposed algorithms. The results have revealed that Greedy-NSGA-II achieves the best performance among our four algorithms, and outperforms three existing non-genetic algorithms. Moreover, the results have also shown that the total bandwidth consumption can be reduced more using our VNF-FG design method and Greedy-NSGA-II together.

Acknowledgements This work was supported in part by National Science Foundation of China (Grant Nos. 61303250, 61302031), Strategic Pilot Project of Chinese Academy of Sciences (Grant No. XDA06010306) and Scientific Research Foundation of the Institute of Information Engineering, Chinese Academy of Sciences (Grant No. Y6Z0011105).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 NFV ISG. Network function virtualization white paper. SDN & OpenFlow World Congress. 2014
2 NFV ETSI ISG. Network functions virtualisation (NFV); Use cases. 2013
3 Dong J K, Jin X, Wang H B, et al. Energy-saving virtual machine placement in cloud data centers. In: Proceedings of 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. New York: ACM, 2013. 618–624
4 Shi W M, Hong B. Towards profitable virtual machine placement in the data center. In: Proceedings of 4th IEEE International Conference on Utility and Cloud Computing. Piscataway: IEEE, 2011. 138–145
5 Pietri I, Sakellariou R. Mapping virtual machines onto physical machines in cloud computing: a survey. ACM Comput Surv, 2016, 49: 49
6 Fischer A, Botero J F, Beck M, et al. Virtual network embedding: a survey. IEEE Commun Surv Tutor, 2013, 15: 1888–1906
7 Ye Z L, Cao X J, Wang J P, et al. Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization. IEEE Network, 2016, 30: 81–87
8 Moens H, Turck F D. VNF-P: a model for efficient placement of virtualized network functions. In: Proceedings of 10th IEEE International Conference on Network and Service Management. Piscataway: IEEE, 2014. 418–423
9 Mehraghdam S, Keller M, Karl H. Specifying and placing chains of virtual network functions. In: Proceedings of 3rd IEEE International Conference on Cloud Networking. Piscataway: IEEE, 2014. 7–13
10 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 1st IEEE Conference on Network Softwarization. Piscataway: IEEE, 2015. 1–9
11 Sahhaf S, Tavernier W, Rost M, et al. Network service chaining with optimized network function embedding supporting service decompositions. Comput Netw, 2015, 93: 492–505
12 Clayman S, Maini E, Galis A, et al. The dynamic placement of virtual network functions. In: Proceedings of IEEE Network Operations and Management Symposium. Piscataway: IEEE, 2014. 1–9
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/IEEE International Symposium on Integrated Network Management. Piscataway: IEEE, 2015. 98–106
14 Mijumbi R, Serrat J, Gorricho J, et al. Network function virtualization: state-of-the-art and research challenges. IEEE Commun Surv Tutor, 2015, 18: 236–262
15 Fonseca C M, Fleming P J. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Proceedings of 5th International Conference on Genetic Algorithms. San Francisco: Morgan Kaufmann Publishers Inc., 1993. 416–423
16 Deb K, Agrawal S, Pratap A, et al. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Proceedings of International Conference on Parallel Problem Solving from Nature. Berlin: Springer-Verlag, 2000. 849–858
17 Sun Q Y, Lu P, Lu W, et al. Forecast-assisted NFV service chain deployment based on affiliation-aware vNF placement. In: Proceedings of IEEE Global Communications Conference. Piscataway: IEEE, 2016. 1–6