An Optimized Deployment Mechanism for Virtual Middleboxes in NFV- and SDN-Enabling Network

Gang Xiong[1]*, Penghao Sun[1], Yuxiang Hu[1], Julong Lan[1] and Kan Li[2]
[1] National Digital Switching System Engineering and Technological R&D Center, Zhengzhou 450002, China
[2] Xi'an Communication Institute, Xi'an 710106, China
[Email: xg1226@126.com, sphshine@126.com, chxachxa@126.com, ndscljl@163.com, studying2012@126.com]
*Corresponding author: Gang Xiong

Received November 12, 2015; revised April 26, 2016; accepted June 17, 2016; published August 31, 2016

Abstract

Network Function Virtualization (NFV) and Software Defined Networking (SDN) are recently considered as very promising drivers of the evolution of existing middlebox services, which play intrinsic and fundamental roles in today's networks. To address the virtual service deployment issues that caused by introducing NFV or SDN to networks, this paper proposes an optimal solution by combining quantum genetic algorithm with cooperative game theory. Specifically, we first state the concrete content of the service deployment problem and describe the system framework based on the architecture of SDN. Second, for the service location placement sub-problem, an integer linear programming model is built, which aims at minimizing the network transport delay by selecting suitable service locations, and then a heuristic solution is designed based on the improved quantum genetic algorithm. Third, for the service amount placement sub-problem, we apply the rigorous cooperative game-theoretic approach to build the mathematical model, and implement a distributed algorithm corresponding to Nash bargaining solution. Finally, experimental results show that our proposed method can calculate automatically the optimized placement locations, which reduces 30% of the average traffic delay compared to that of the random placement scheme. Meanwhile, the service amount placement approach can achieve the performance that the average metric values of satisfaction degree and fairness index reach above 90%. And evaluation results demonstrate that our proposed mechanism has a comprehensive advantage for network application.

***Keywords*:** Software-defined networking; network function virtualization; middlebox; quantum genetic algorithm; game theory

This work is supported by The National Basic Research Program of China (973) (Grant No. 2012CB315901, 2013CB329104), The National Natural Science Foundation of China (Grant No. 61309019, 61372121, 61572519, 61502530), and The National High Technology Research and Development Program of China (863) (Grant No. 2015AA016102, 2013AA013505).

1. Introduction

Current networks rely on the rich functionality introduced by a wide spectrum of specialized appliances or middleboxes [1]. Taking **Fig. 1** for example, network address translation (NAT) and load balancers improve critical performance, firewalls and intrusion detection systems (IDS) ensure network security, content filters and wide area network (WAN) optimizers reduce bandwidth costs. Research [2] shows that the number of middleboxes is on par with the number of routers in enterprise networks, such as an average very large network that hosts about 2,850 Layer 3 routers and 1,946 total middleboxes. Furthermore, it is reported that the market for network security appliances alone is estimated to rise to ten billion in 2016 [3]. So it is believed that the number of middleboxes as a critical part of today's networks will be on rapid growth in the foreseeable future.

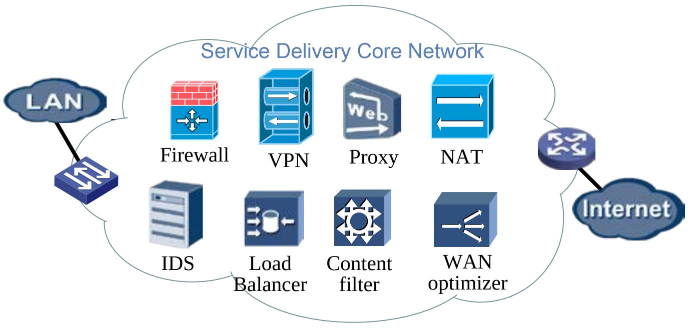


Fig. 1. An appliance site of middlebox in service delivery network

Middleboxes are inevitably deployed in enterprise networks, data centers and cloud environments and so on. They play critical roles in introducing new network functionality into the networks. However, it is annoying that such proprietary middleboxes come with a number of significant drawbacks [2]: (1) it is expensive to buy and manage middleboxes which leave potential small players out of the market and also raise innovation barriers. (2) Introducing hardware appliances for new network features usually needs a long deployment process which on average takes four years. (3) Acquired from independent vendors and deployed as standalone devices, middleboxes lack hooks and uniform APIs for extension or experimentation. (4) The waste of resources is produced due to the fact that hardware middleboxes cannot be scaled up and down easily with shifting demand.

It has received a significant amount of attention on how to solve these issues of middleboxes. In a general context, the recent solution strategies are built on similar reasoning that combine the benefits of two new networking technologies, namely Network Function Virtualization (NFV) [4] and Software Defined Networking (SDN) [5]. These two concepts have emerged aiming at cost reduction, increment of network scalability and service flexibility with the strategies that can enable innovation in network nodes (e.g.

switches and routers), for example, standardized APIs, software-centric implementations and commodity hardware. NFV proposes to run the network functions as software instances on commodity servers or datacenters, while SDN supports a decomposition of networks into control-plane and data-plane functions. Therefore, these new concepts are considered as very promising drivers to design cost-efficient middlebox service architectures [6].

However, to some extent, due to the difference of middleboxes from routers and switches, applying these new ideas to middleboxes in networks raises unique challenges and opportunities. For example, introducing NFV to networks requires transportation of the network data traffic to the demanded network services, which imposes additional transport delay on networks [7]. In addition, consolidating multiple heterogeneous virtualized middlebox modules on a shared service platform also raises new challenges for resource allocation. Therefore, in order to guarantee the overall network utilization, the objective of this paper is to define, model and solve the problem of how to optimally deploy virtual middlebox services in the network environment enabling NFV and SDN. We make the following three key contributions:

- First, for the sub-problem one that involves service location placement, we model the question by the integer linear programming and develop a heuristic solution based on the improved quantum genetic algorithm.
- Second, for the sub-problem two that involves service amount placement, we explore a cooperative game based mathematical model and a distributed placement algorithm which achieves the Nash Bargaining Solution (NBS) for sharing network resources.
- Third, we demonstrate the performance of our proposed solution through extensive simulations under different experimental scenarios.

Roadmap: The rest is organized as follows: Section 2 summarizes the related works; Section 3 defines the problems that we solve in this paper; Section 4 proposes our solution; Section 5 evaluates and analyzes the algorithms performance; Section 6 concludes this paper.

2. Related Work

A growing amount of researches focus on designing schemes that are amenable for the evolution of the middlebox service model. Broadly speaking, there are mainly two complementary approaches that are pursued. The first tackles the problem of high building capital expenditures and limited extensibility caused by the hardware-based devices, with software-centric service framework. The second tackles the problem of high operation expenditures and limited flexibility in the service procedure, with SDN controlling routing through the specified functional sequence (i.e. Service Function Chaining, SFC). The main works related with these two researches can be summarized as follows.

On the one hand, [2] proposed a practical service framework for outsourcing enterprise middlebox processing to the shared cloud computing platform. [8] enabled innovation in middlebox deployment with software-centric middlebox implementations running on general-purpose hardware platforms. [9] recognized virtual middleboxes as first-class entities and presented a framework for immediate application deployment over- or under-the-cloud. Furthermore, [10] realized a software-defined middlebox networking framework to simplify the management of complex and diverse functionalities. [11] designed a control plane called OpenNF that could provide efficient and coordinated control of both internal middlebox state and network forwarding state.

On the other hand, [12] presented SIMPLE architecture, an SDN-based policy enforcement layer for efficient middlebox-specific traffic steering. Built upon OpenFlow protocol [13], [14] proposed a scalable framework (called StEERING) for dynamically routing traffic through any sequence of middleboxes. [15] developed FlowTags architecture, which consisted of SDN controllers and FlowTags-enhanced middleboxes, to integrate middleboxes into SDN networks. [16] proposed a solution for routing traffic in SDN-enabled dynamic network with consolidated middleboxes implemented using virtual Machines. [17] enhanced the adaptability of network nodes via service function chain construction.

The researches mentioned above either focus on introducing NFV and SDN to design service architectures of middleboxes, or are concerned with traffic steering of middle-box services. Nevertheless, these studies are based on the assumption that middlebox services have been completely deployed. Therefore, limited attention has been drawn to the concrete problem of service placement.

In fact, SDN and NFV provide great flexibility for deployment of middlebox services. And reasonable service approach can better protect the network performance and quality of service (in terms of total throughput, load balancing [18], link utilizations, etc.). Therefore, this paper further studies the optimization of the service deployment method in order to support the existing or future researches.

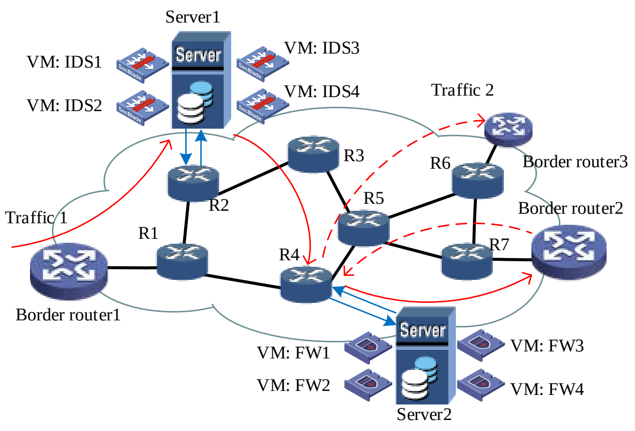
3. Problem Statement

There are lots of advantages in introducing SDN and NFV to the network function, but it also raises some challenges for ensuring the network efficiency, taking **Fig. 2** for example.

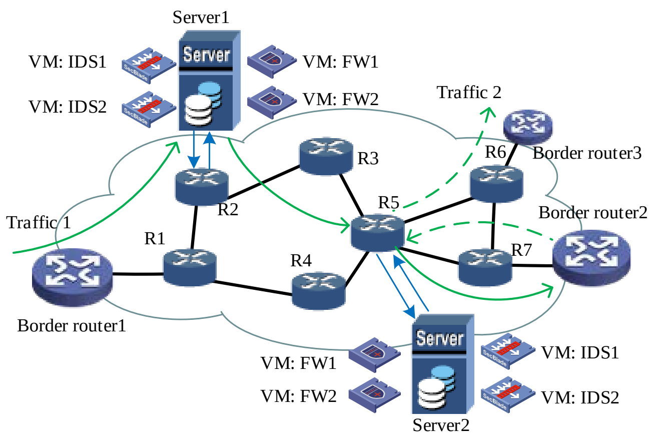
In **Fig. 2**(a), two types of virtual middlebox (VM), i.e. IDS and Firewall (FW) operating on general server 1 and server 2, are placed at the network node R2 and R4. We assume that traffic 1 which requires IDS and FW services enters the network from border router 1 and exits the network on border router 2, while traffic 2 needing the FW service enters the network from border router 2 and exits the network on border router 3. However, since server 1 only supports the IDS function and server 2 solely operates the FW function, traffic 1 has to traverse the IDS service in R2 and then traverse the FW service in R4 (shown in the red solid curve), and traffic 2 must be steered to the FW service in server 2 (shown in the red dash curve). Apparently, the original placement scheme concerning the location and amount of the VMs increases traffics delay and network resource cost.

In Fig. 2(b), we adjust the deployment policy, which makes both server 1 and 2 operate IDS and FW VMs, and migrates server 2 from R4 to R5. Under this placement scenario, the requirement of traffic 1 can be completely responded by server 1 and the requirement of traffic 2 is also satisfied at server 2, which saves transportation time and link cost (respectively, shown in the green solid and dash curve).

Thus, given the ability to flexibly place the middlebox service in the NFV+SDN scenarios, we discover that certain placement strategies are better than others when concerning network performance (load, delay, etc.). We name the problem of how to find the optimal deployment strategy for virtual middleboxes as the service deployment problem.



(a) The original scheme



(b) The adjusted scheme

Fig. 2. Motivating examples of the middlebox deployment problem

Usually, it is difficult to uniformly define the service deployment problem, which is both comprehensive and complex. In practical application, a specific deployment strategy should be established according to specific application scenario. In this paper, for supporting the scenario of cost-efficient service framework and service chain（mentioned in section 2）,we discuss the service placement problem which focuses on transport delay and load balance. Based on our analysis, transport delay relates to the location of the services placement, while the load balance relates to the amount of the service placement, which is specifically explained as follows:

***Sub-problem 1 being location of the services*:** Each middlebox occurring at a small set (1 or 2) of the hops that a packet traverses processes only a subset of packets pertinent to its application function. Apparently, if the middleboxes are deployed randomly or at some remote nodes, the network traffic is sent on a detour through the middlebox services and leads to a potential increase of latency (**Fig. 2(a)** red curve). Thus, there is still an orthogonal problem of where these middlebox services are deployed so that this performance penalty is minimized.

***Sub-problem 2 being amount of the services*:** A software-centric service policies are enforced by configuring heterogeneous virtual middleboxes on a shared platform. However, with the limitation of platform resource (such as computation, memory), consolidating various workloads on a shared platform raises new challenges for resource allocation to retain the network performance, including service requirements, allocation fairness, etc. Therefore, there is also another urgent problem of how many different virtual middleboxes are instantiated in each platform so that the service performance is optimal.

The two sub problems only propose a specific representation of the initial service deployment problem, and maybe there are different representations in other researches. To the best of our knowledge, [19] [20] discuss the sub-problem one for the mobile core network, but their contributions cannot be directly used for virtual middleboxes placement. Even more, they pay little attention to resource allocation.

Thus, our focus in this paper is more on highlighting the middlebox deployment problem. First, we explore where to place services in networks with the objective of minimizing average transmission time for subscribers' traffics; then we address the problem of how to allocate network resource to heterogeneous services to achieve the load balance among soft instances. We attempt to solve these two problems to find a reasonable solution for the initial middlebox service problem.

4. Proposed Solution

In this section, we propose our solution for the problems from the previous section, which is an Optimal Middlebox Deployment policy Maker, called OMDM. Our OMDM can decide reasonable high-level deployment policies through the network topology and resource, and translate this policies into an efficient and load balanced configuration in data plane.

4.1 System overview

Fig. 3 gives an overview of OMDM architecture where virtual implementations of middleboxes are consolidated to run on general-purpose shared hardware platforms and are managed in a logically centralized manner with uniform APIs for a network-wide view. This SDN and NFV-based solution reduces the cost and development cycles to build and deploy new middlebox applications. As illustrated in **Fig. 3**, OMDM architecture consists of various components, which can be classified into three kinds: (1) The control plane components including the network operation system and control modules. (2) The data plane components containing OpenFlow switches and virtual middleboxes. (3) The interfaces between control plane and data plane. Next, we describe these roles of main components.

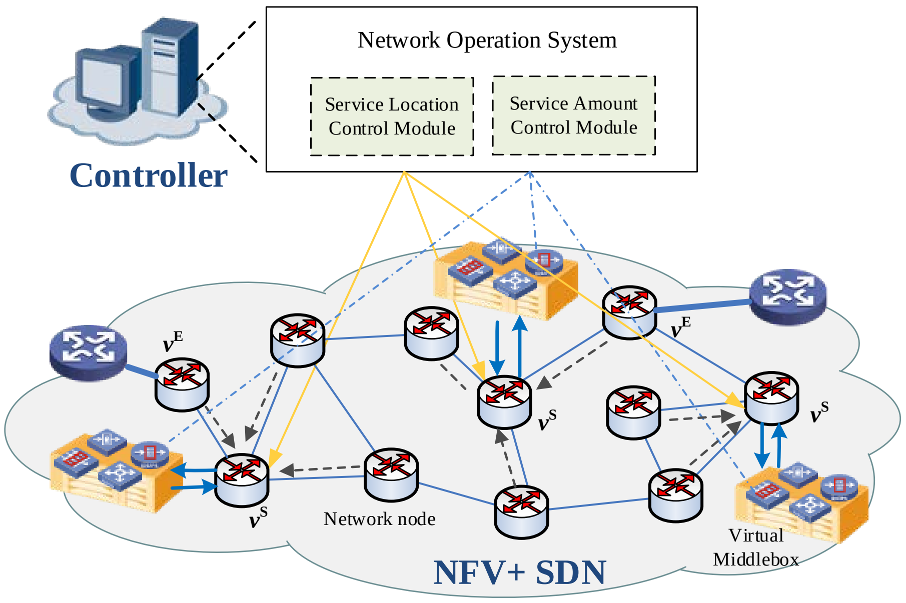


Fig. 3. Overview of OMDM for middlebox deployment

The controller is centric administrator of the network and plays a core role in our proposed scheme. The controller takes three inputs: (i) the network topology; (ii) for each middlebox, a description of its function and hardware resources such as CPU, memory, bandwidth; (iii) the network-wide traffic workload in the form of a traffic matrix.

In the physical network (i.e. the data plane [21]), these virtual middleboxes run upon some general platforms which are connected with the network nodes (e.g. open switch). Then under the steering of forwarding rules, the traffics pass through nodes and some middlebox services.

The uniform APIs interfaces (such as OpenFlow protocols [22]) are used for communication between the controller and network nodes, such as network information collection and deployment policy configuration.

The process procedure of OMDM scheme is as follows: with the input databases in controller and the requirement of placement, the controller runs two control modules, i.e. service location control module and service amount control module, which solve an optimization problem of service placement while respecting traffic transmission and resource limits. First, the location control module runs an algorithm to determine the best locations where middleboxes can provide the network service for these traffics from nodes without placing the middleboxes (as shown the black dashed lines in **Fig. 3**). Second, the amount control module runs another algorithm that determines the placement amount of each middlebox on selected network nodes. Finally, the results of two modules are output for configuring the virtual network middlebox services by the uniform APIs.

In subsequent section, we focus on designing the control policies of the middlebox service location and amount modules.

4.2 Service location decision

4.2.1 Integer linear programming model

We formulate the service location selection problem as an optimization problem of minimizing the delay or distance to be traversed by all subscribers' traffics. Assume that the network topology is defined by an undirected graph $G=\left(V,E\right)$, with node set V and edge set E, for example the topology of **Fig. 3**. G is a symmetric graph with weighted edges and each edge is associated with delay $d\left(l\right),\ l\in E$.

The objective is to find a subset $V^{\mathrm{S}}\ (V^{\mathrm{S}}\subset V\ )$ of service locations among all $N\left(\left|V\right|=N\right)$ candidates so that the total delay for all users are minimized. The optimization problem can be considered as an Integer Linear Programming (ILP) model, as follows:

Firstly, the minimum delay between two nodes $v_i, v_j\ \left(v_i, v_j\in V; v_i\neq v_j\right)$, is calculated by:

$$d\left(v_i,v_j\right)=\arg\min_{p_t\in P\left(v_i,v_j\right)}\sum\nolimits_{l\in p_t} d\left(l\right),\ \ \left(i,j=1,2,\ldots,N\right) \tag{1}$$

where $P\left(v_i,v_j\right)$ denotes the set of path from node v_i to node v_j, p_t is one path element of the set $P\left(v_i,v_j\right)$. Let $\mathbf{D}=\left[d\left(v_i,v_j\right)\right]_{N\times N}$ denote the shortest-path matrix of graph G.

Secondly, the variables of this optimization problem are $x_i\in\left\{0,1\right\}$ with zero value corresponds to the node v_i that has not been selected for service placement, and unit value is used to denote that v_i is the location of network service. And $V^{\mathrm{E}}\left(V^{\mathrm{E}}\subset V\right)$ is the egress node set. The linear optimization model is shown as:

$$\min\sum_{i=1}^{N} x_i d\left(v_i,v_i^{\mathrm{E}}\right)+\sum_{i=1}^{N}\left(1-x_i\right)d\left(v_i,v_i^{\mathrm{S}}\right), \tag{2}$$

$$s.t.\ \ v_i^{\mathrm{S}}=\{v_n\mid v_n\in V, x_n=1, d(v_i,v_n)=\arg\min_{j\in N, x_j=1} d(v_i,v_j)\}, \forall v_i\in V, \tag{3}$$

$$v_i^{\mathrm{E}}=\{v_k\mid v_k\in V^{\mathrm{E}}, d(v_i,v_k)=\arg\min_{v_j\in V^{\mathrm{E}}} d(v_i,v_j)\}, \forall v_i\in V, \tag{4}$$

$$x_i=\left\{0,1\right\}, i=1,2,\ldots,N, \tag{5}$$

where Eq. (2) defines the total delay that is calculated as the sum of the delay between ingress points and service points, plus and the delay between service points and egress points. v_i^{S} is the service node responding to node v_i, and Eq. (3) means to assign the service node with minimum delay to v_i as its v_i^{S}. v_i^{E} is the egress point responding to node v_i, and constraint (4) says that the egress node with minimum delay to v_i is selected as its v_i^{E}.

4.2.2 Solving algorithm for ILP model

The basic quantum genetic algorithm (QGA) has a good performance in dealing with integer programming problem [23]. Furthermore, we design the Improved Quantum Genetic Algorithm (IQGA) with some improving measures, such as the dynamic rotation angle and quantum mutation. Then, we propose a solving algorithm (called SLP-IQGA) to obtain the optimal Service Location Policy (SLP). The main steps of SLP-IQGA algorithm are described by:

Step 1: Getting the shortest path matrix $\mathbf{D}$. The Matrix $\mathbf{D}$ is an important input parameter, and contains all minimum delay of any two nodes in graph G. The element $d\left(v_i, v_j\right)$ in $\mathbf{D}$ can be calculated by running Bellman-Ford algorithm.

Step 2: Initialization of quantum genetic algorithm. The basic unit in quantum computation is the qubit. A qubit can be represented by a superposition of the basis states $\left|0\right\rangle$ and $\left|1\right\rangle$:

$$\left|q\right\rangle = \alpha\left|0\right\rangle + \beta\left|1\right\rangle, \alpha^2 + \beta^2 = 1, \tag{6}$$

where α, β is a complex number, respectively, denoting the probability amplitude of the basis states. Each feasible solution of QGA is the element chromosome, which is made up of multiple qubits. A chromosome containing N qubits is described by:

$$C = \left[\begin{array}{c|c|c|c} \alpha_1 & \alpha_2 & \cdots & \alpha_N \\ \beta_1 & \beta_2 & \cdots & \beta_N \end{array}\right], \tag{7}$$

In SLP-IQGA algorithm, the qubit q_i $\left(i = 1, 2, \ldots, N\right)$ of chromosome C is the $\left|1\right\rangle$ state means that node v_i is chosen as the service placement location, otherwise node v_i is not placed service. At the initial time, we set the population size M and denote t_{th} generation population $\mathcal{P}\left(t\right) = \left\{C_1^{(t)}, \ldots, C_m^{(t)}, \ldots, C_M^{(t)}\right\}$ where $C_m^{(t)}$ is described in Eq. (7). We set all states appear in the same probability, i.e. $\alpha_{mi}^{(0)} = \beta_{mi}^{(0)} = 1/\sqrt{2}$.

Step3: Measuring the observation value of chromosome C. The chromosome observation is to make each qubit of chromosome collapse into a certain state. The measurement method is to generate a random number among the range [0, 1] for each qubit, if the random number is less than $\left|\alpha\right|^2$, the measurement value x of the qubit is 0, otherwise as 1. After the measurement operation, chromosome C is transformed to the observation value $X_C = \left\{x_1, x_2, \ldots, x_N\right\}$.

Step4: Fitness calculation. The fitness is the standard to measure the quality of the individual. The higher the fitness is, the closer the individual is to the optimal solution. For individual $X_C = \left\{x_1, x_2, \ldots, x_N\right\}$, the fitness function can be obtained by Eq. (8) as follows:

$$Fit\left(X_C\right)=\left[\sum_{i=1}^{N}x_i d\left(v_i,v_i^{\mathrm{E}}\right)+\sum_{i=1}^{N}\left(1-x_i\right)d\left(v_i,v_i^{\mathrm{S}}\right)\right]^{-1}. \tag{8}$$

Step5: Adaptive adjustment strategy of quantum rotation gate. In quantum genetic algorithm, the population can be updated by quantum rotation gate $\boldsymbol{U}\left(\theta\right)$. Based on the quantum rotation gate, the adjustment operation of the i_{th} qubit of $C_m^{(t)}\left(C_m^{(t)}\in\mathcal{P}\left(t\right)\right)$ is as follows:

$$\begin{bmatrix}\alpha'_{mi}\\ \beta'_{mi}\end{bmatrix}=\boldsymbol{U}\left(\theta\right)\begin{bmatrix}\alpha_{mi}\\ \beta_{mi}\end{bmatrix}=\begin{bmatrix}\cos\theta_i & -\sin\theta_i\\ \sin\theta_i & \cos\theta_i\end{bmatrix}\begin{bmatrix}\alpha_{mi}\\ \beta_{mi}\end{bmatrix}\left(i=1,2,\ldots,N\right), \tag{9}$$

where α'_{mi}, β'_{mi} represent the adjusted probability amplitude. θ_i denotes the rotation angle of quantum rotation gate, which is defined by :

$$\theta_i=s(\alpha_i,\beta_i)\cdot\Delta\theta_i, \tag{10}$$

where $s(\alpha_i,\beta_i)$ determines the direction of quantum rotation and $\Delta\theta_i$ determines the size of the quantum rotation. In order to reduce the influence of the rotation angle on the convergence rate of the algorithm, the adaptive method is used to adjust θ_i. Specific adjustment policies are shown in **Table 1**, where δ is a coefficient related to the convergence rate of the algorithm and we set it as a variable changed with the number of iterations:

$$\delta=0.04\pi\left(1-\sigma\times\frac{t}{T+1}\right), \tag{11}$$

where σ is a constant of [0, 1], t is the current iteration, and T is the iteration rounds.

Table 1. Adjustment method of rotation angle

x_i	x_i^{b}	Fit(X)≥Fit(X_{b})	$\Delta\theta_i$	$s(\alpha_i,\beta_i)$			
				$\alpha_i\beta_i>0$	$\alpha_i\beta_i<0$	$\alpha_i=0$	$\beta_i=0$
0	1	False	0	0	0	0	0
0	1	True	δ	-1	±1	±1	0
1	0	False	δ	-1	±1	±1	0
1	0	True	δ	1	-1	0	±1
1	1	False	δ	1	-1	0	±1
1	1	True	δ	1	-1	0	±1

Note: X_{b} is the current optimal solution, x_i^{b} is the i_{th} qubit of X_{b}.

Step6: Quantum variation and quantum crossover. The quantum variation may generate new individuals in order to prevent QGA evolving into the local optimal solution. We choose a small ratio of the population, appoint randomly a variable qubit of the chromosomes, and swap the probability amplitude α,β in the appointed qubit. On the other hand, quantum crossover can produce more new individuals to improve the algorithm search ability. Our specific implementation is that all individuals in the population are ordered randomly and then a new population is obtained by cyclically shifting *i*-1 number of the *i* bit in all ordered individuals.

Based on the above description, the service location module runs SLP-IQGA algorithm and obtain the result $X_{\mathrm{b}}=\left\{x_1,x_2,\ldots,x_N\right\}$, where $x_i=1$ represents node v_i being the service location v^S. The whole process flow is shown in **Table 2**.

Table 2. The process flow of SLP-IQGA algorithm

```
Input:   Network topology graph $G$, Egress node set $V^E$, Population size $M$,
         Chromosomes length $N$,   Variation ratio $r$,   Rounds of evolution iteration $T$.
Output:  Optimal placement location scheme $\boldsymbol{X}_b$.
1.  Run the Bellman-Ford algorithm for calculating the shortest path matrix $\boldsymbol{D}$ of $G$;
2.  Generate the original population $\mathcal{P}(t)=\{C_1^{(t)},\dots,C_M^{(t)}\}$ ($t$←0) and initialize each individual;
3.  Measure the $\mathcal{P}(t)$ and get the observation value $O(t)=\{X_1,X_2,\dots,X_M\}$;
4.  for all $\boldsymbol{X}_i \in O(t)$ do
5.      for all $x_j \in \boldsymbol{X}_i$ do
6.          if $x_j=0$ then
7.              Query matrix $\boldsymbol{D}$, assign a minimum delay node $v_i^S$ to $x_j$;
8.      end for
9.      Calculate the fitness value Fit($\boldsymbol{X}_i$);
10. end for
11. $\boldsymbol{X}_b$={ $\boldsymbol{X}_i$ |Fit($\boldsymbol{X}_i$)=arg max $_{X\in O(t)}$ Fit($\boldsymbol{X}$)};
12. while($t<T$)
13.     Evolve $\mathcal{P}(t)$ to $\mathcal{P}(t+1)$ by quantum rotation gate, $t=t+1$;
14.     Do the process steps from 3 to 10;
15.     $B(t)$ ={ $\boldsymbol{X}_i$ |Fit($\boldsymbol{X}_i$)=arg max $_{X\in O(t)}$ Fit($\boldsymbol{X}$)};
16.     if Fit($\boldsymbol{X}_b$)< Fit($B(t)$)then
17.         $\boldsymbol{X}_b = B(t)$ ;
18.     if ($\boldsymbol{X}_b$ has not changed after N/2 evolution generation) then
19.         Do quantum crossover and quantum variation for $\mathcal{P}(t)$ ;
20. end while
```

Generally, the service location problem is a NP hard problem whose complexity grows exponentially with the number of network nodes. By analyzing algorithm flow, we can obtain that the computational complexity of SLP-IQGA algorithm is just $O(M\times N)$, i.e. $O(n^2)$. Therefore, the proposed algorithm can effectively reduce the complexity of the original problem by using the heuristic approach of improved QGA.

4.3 Service amount decision

For deciding the service deployment amount, we derive a cooperative game model by taking advantage of the Nash bargaining which has been widely used to balance the tradeoff between fairness and efficiency for resource sharing problems [24].

4.3.1 Cooperative game model

We assume that there are J general service platforms (such as servers) $\mathcal{G}=\{g_1,g_2,\dots,g_J\}$ connected with respective nodes and K types of middlebox services $\mathcal{S}=\{s_1,s_2,\dots,s_K\}$ in a network, as shown in **Fig. 4**.

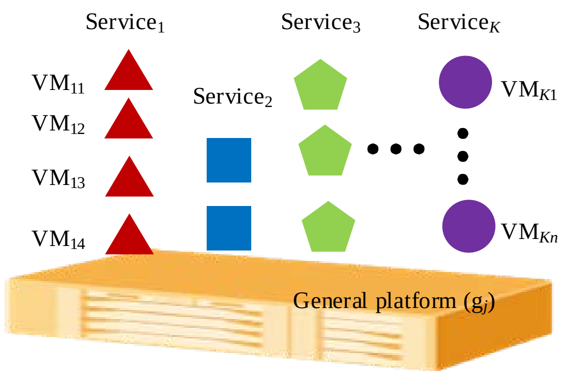


Fig. 4. The illustration of virtual services deployment amount

The service amount is the number of corresponding virtual middlebox instances, and we use the symbol $n_{kj}(t)$ expressing the instance number of service s_k on platform g_j at time *t*. The network resource capacity Z_j in platform g_j is represented by the maximum VM instance number that it can support, and we get the capacity constraint vector $\mathcal{Z}=\{Z_1,Z_2,\ldots,Z_J\}$. The service requirement in each platform is described by the matrix $\mathbf{R}(t)=\left[R_{kj}(t)\right]_{K\times J}$, where $R_{kj}(t)$ denotes the demand of service s_k on platform g_j at time *t*. So the amount decision problem means that under the service requirement, how to find the optimal scheme $n_j^*(t)=\{n_{1j}^*(t),n_{2j}^*(t),\ldots,n_{Kj}^*(t)\}$ in the feasible field. Because we discuss instantaneous demand at a specific time, the variable *t* is dropped for simplicity as follows.

According to the game theory, the resources allocation problem above can be modeled into a game problem where the *K* types of service can be viewed as players who are competing for their platform resources. Since the space of service deployment is a nonempty convex closed and upper-bounded set, there is a Nash Bargaining Solution (NBS) for the game problem [25]. We next define the optimization problem which aims to achieve the NBS.

The n_{kj} is what we need to deploy the service s_k on platform g_j. The deployment is bounded by the physical resources of each platform, i.e. $\sum_{k=1}^{K} n_{kj}(t)\le Z_j$.

To guarantee the base deployment amount, we use the constraint $n_{kj}\ge L_{kj}$ to ensure that each service can obtain an initial minimum amount, where

$$L_{kj}=\min\{B_k,R_{kj}\}. \tag{12}$$

B_k represents the base amount for the service s_k and $R_{kj}\left(R_{kj}\in\mathbf{R}\right)$ is the requirement amount which needs to be specified by using a simple method as follows:

$$R_{kj}=TR_{kj}/e_{kj}, \tag{13}$$

where TR_{kj} is the number of network traffics demanding service s_k and e_{kj} denotes the number of traffics processed by a single service instance.

In addition, U_{kj} is the upper bound for deployment amount, which is denoted by

$$U_{kj} = \min\left\{Z_j, R_{kj}\right\}. \tag{14}$$

After determining the lower and upper amounts for each service via Eq. (12) and (14), we have the joint profit optimization problem (P_n) as follows:

$$(P_n) \quad \max_n \sum_k \sum_j \ln\left(n_{kj} - L_{kj}\right), \tag{15}$$

$$s.t. \quad n_{kj} \geq L_{kj},\ \forall k \in \{1,\dots,K\}, j \in \{1,\dots,J\}, \tag{16}$$

$$n_{kj} \leq U_{kj},\ \forall k \in \{1,\dots,K\}, j \in \{1,\dots,J\}\ , \tag{17}$$

$$\sum\nolimits_k n_{kj} \leq Z_j,\ \forall j \in \{1,\dots,J\}. \tag{18}$$

Note that the variable constraints are linear, the Kuhn-Tucher conditions are necessary and sufficient for an existing optimal solution. Next, we apply the method of Lagrange multipliers to derive the formal optimal solution.

Theorem 1: There exists Lagrange coefficient $\gamma_j \geq 0\left(j \in \{1,\dots,J\}\right)$ such that

$$n_{kj}^* = L_{kj} + \frac{1}{\sum_j \gamma_j I_{kj}},\ \forall k \in \{1,\dots,K\}, j \in \{1,\dots,J\}, \tag{19}$$

where n_{kj}^* is the unique NBS for the optimization problem P_n, the $I_{kj} \in \{0,1\}$ is a binary variable, where $I_{kj} = 1$ means s_k is placed on g_j, and $I_{ik} = 0$ is opposite.

Proof: Assume that the solution space $\mathcal{Y} \subset \mathcal{R}^{K\times J}$ is a nonempty, convex and compact set. We define

$$\psi(n) = \sum_{k=1}^{K}\sum_{j=1}^{J} \ln\left(n_{kj} - L_{kj}\right), \tag{20}$$

then $\psi(n)$: $\mathcal{Y} \to \mathcal{R}^+$ is strictly concave. Let $\lambda_{kj} \geq 0$, $\eta_{kj} \geq 0$, and $\gamma_j \geq 0$ denote the Lagrange multipliers, respectively. Then, the Lagrange equation of problem P_n is

$$\begin{aligned}\mathcal{L}(n,\lambda,\eta,\gamma) = \psi(n) &- \sum_{k=1}^{K}\sum_{j=1}^{J} \lambda_{kj}\left(L_{kj} - n_{kj}\right) \\ &- \sum_{k=1}^{K}\sum_{j=1}^{J} \eta_{kj}\left(n_{kj} - U_{kj}\right) - \sum_{j=1}^{J} \gamma_j\left(\left(\overrightarrow{n_j} \times \overrightarrow{I_j}\right) - Z_j\right)\end{aligned}, \tag{21}$$

where $\overrightarrow{n_j} = [n_{1j},\dots,n_{Kj}]$, $\overrightarrow{I_j} = [I_{1j},\dots,I_{Kj}]$ $\left(j \in \{1,\dots,J\}\right)$. To give the necessary and sufficient conditions for optimizing the objective $\partial\mathcal{L}\left(n^*,\lambda,\eta,\gamma\right)/\partial n = 0$, we have

$$\frac{1}{\left(n_{kj}^* - L_{kj}\right)} + \lambda_{kj} - \eta_{kj} - \sum_{j=1}^{J} \gamma_j I_{kj} = 0,\ \left(\forall k \in \{1,\dots,K\}, j \in \{1,\dots,J\}\right). \tag{22}$$

And with the

$$\begin{cases}\lambda_{kj}\left(L_{kj} - n_{kj}^*\right) = 0,\ \eta_{kj}\left(n_{kj}^* - U_{kj}\right) = 0,\ \gamma_j\left(\sum_{k=1}^{K} n_{kj}^* - Z_j\right) = 0, \\ \forall k \in \{1,\dots,K\}, j \in \{1,\dots,J\}.\end{cases} \tag{23}$$

where $n^* = \left(n_{11}^*,\dots,n_{KJ}^*\right)$ is the optimal solution to the problem P_n. To derive the solutions, we should consider the values of the multipliers in Eq. (23). Focusing on such a general situation that $L_{kj} \leq n_{kj} \leq U_{kj}$, which can derive $\lambda_{kj} = 0$ and $\eta_{kj} = 0$. Then we can obtain the expression of the n_{kj}^* by solving Eq. (22). Here, the proof is completed.

In summary, we have shown how to achieve the centralized solution of the deployment problem. However, the computational complexity of the original problem P_n may increase significantly as the number of service types and service nodes scales up. This motivates us to explore the distributed solving manner.

4.3.2 Solving algorithm for game model

In this section, we present a Service Amount Policy (SAP) approach based on a Distributed Cooperative Game Algorithm (DCGA), called SAP-DCGA.

First of all, we consider the primal problem $\overline{P_n}$ which has the same solution as problem P_n:

$$\left(\overline{P_n}\right) \quad \min_n -\sum_k \sum_j \ln\left(n_{kj} - L_{kj}\right), \tag{24}$$

where the variable constraints of problem $\overline{P_n}$ are same to P_n, i.e. Eq. (16)-Eq. (18).

We mainly discuss the general situation that $L_{kj} \le n_{kj} \le U_{kj}$. The Lagrangian associated with the $\overline{P_n}$ is defined as $\mathcal{L}(\bullet): \mathcal{Y} \times \mathcal{R}^J \to \mathcal{R}$, where

$$\mathcal{L}(n,\gamma) = -\sum_{k=1}^{K}\sum_{j=1}^{J} \ln\left(n_{kj} - L_{kj}\right) + \sum_{j=1}^{J} \gamma_j \left(\left(\overrightarrow{n_j} \times \overrightarrow{I_j}\right) - Z_j\right). \tag{25}$$

Note that γ is the dual variables associated with the problem. The Lagrange dual function $\mathcal{D}(\bullet): \mathcal{R}^J \times \mathcal{R}^J \to \mathcal{R}$ corresponding to $\mathcal{L}(n,\gamma)$ is expressed as

$$\mathcal{D}(\gamma) = \inf_{n \in \mathcal{R}^{K \times J}} \mathcal{L}(n,\gamma). \tag{26}$$

Since $\mathcal{Y}$ is convex and P_n is convex over $\mathcal{Y}$, there is no duality gap and there exists γ satisfying $\mathcal{D}(\gamma) = \mathcal{L}(n^*,\gamma)$. The optimal solution γ can also be derived through the dual problem $P_{\mathcal{D}}$ corresponding to the primal problem with no duality gap, which is shown as:

$$\left(P_{\mathcal{D}}\right) \quad \max_{\gamma \in \mathcal{R}^J} \mathcal{D}(\gamma) = \mathcal{L}(n^*,\gamma). \tag{27}$$

To solve the dual problem, we design an algorithm that converges to the optimal γ in the gradient projection method. We define the following recursion

$$\gamma_j^{(w+1)} = \gamma_j^{(w)} + \varepsilon \frac{\partial \mathcal{D}}{\partial \gamma_j} \ \forall j \in \{1,\ldots,J\}, \tag{28}$$

where ε is the step-size, and the partial derivatives of $\mathcal{D}(\gamma)$ is obtained from the Eq. (25) by replacing n_{kj}^* as

$$\frac{\partial \mathcal{D}}{\partial \gamma_j} = \left(\overrightarrow{n_j} \times \overrightarrow{I_j}\right) - Z_j = \sum_k I_{kj} n_{kj}^* - Z_j, \ \forall j \in \{1,\ldots,J\}. \tag{29}$$

It can be proved that if $\gamma_j^{(0)} \in \mathcal{R}^J$ and $\varepsilon \in \left(0, 2/\kappa\right]$ (κ is the Lipschitz constant), then the recursive sequence $\left\{\gamma_j^{(w)}\right\}$ is convergent, i.e. $\lim_{w \to \infty} \gamma_j^{(w)} = \gamma_j^*$.

In Theorem 1, we have obtained the explicit form of optimal n_{kj}^*. Since there is no duality gap in the dual decomposition, the amount n associated with γ converges to the NBS, thus

$$\lim_{w\to\infty} n_{kj}\left(\gamma_j^{(w)}\right) = n_{kj}^* . \tag{30}$$

Based on above guidelines, we demonstrate the algorithm flow of SAP-DCGA in Table 3.

Table 3. The process flow of SAP-DCGA algorithm

```
Input:   Platform capacity Z_j, Service base amount B_k, Requirement matrix R=[R_kj],
         Rounds of iteration W, Step-size ε.
output:  Optimal service placement amount scheme $\left[n_{kj}\right]_{K\times J}$.

1.  Calculate the deployment matrix $\left[I_{kj}\right]_{K\times J}$ based on the matrix R;
2.  for all $n_{kj}$ do
3.        Initialize the lower and upper bounds of variable, $L_{kj}$ and $U_{kj}$
4.        $n_{kj} = L_{kj}$
5.  end for
6.  while(steps $w<W$) do
7.     for all service platform $g_j$ do
8.        Update $\gamma_j^{(w)} = \gamma_j^{(w-1)} - \varepsilon\left(Z_j - \sum_k I_{kj} n_{kj}^{(w-1)}\right)$;
9.     end for
10.    for all $n_{kj}$ do
11.       If $1/\gamma_j^{(w)} \ge U_{kj} - L_{kj}$ then $n_{kj}^{(w)} = U_{kj}$;
12.       else $n_{kj}^{(w)} = L_{kj} + 1/\gamma_j^{(w)}$;
13.    end for
14.    do $w$++
15. end while
```

Similarly, the complexity of service amount problem increases exponentially with the number of service types and platforms when an exhaustive search is used to solve it. However, the computational complexity of SLP-IQGA algorithm is just $O(K\times J)$, i.e. $O(n^2)$. Therefore, our algorithm can effectively improve the efficiency of the solving computation.

5. Performance Evaluation

To evaluate the performance of our algorithms, we set up the experiment environment on a computer equipped with an Intel(R) Core(TM) i7 CPU 2.67GHz processor with 2 cores, and 4GB of RAM. The GT-ITM Tool [26] is used for generating different network topologies and the proposed algorithms are implemented by MATLAB software.

5.1 Evaluation results of SLP-IQGA algorithm

We evaluate the SLP-IQGA algorithm in section 4.2 using the test topology from GT-ITM and simulated real network traffics from data center network traffic record [27]. The method of generating links is *Waxman*, with the parameters *alpha*=0.3, *beta*=0.2. The link delay in test topology is measured in unit of millisecond and the delay value of each link is randomly in the range of [1, 100]. In SLP-IQGA, we set the population size M=20 and the variation probability r= 0.1.

5.1.1 Effectiveness of parameters configuration

With the intelligent decision, SLP-IQGA can calculate automatically the service node number (N_S) under different network topologies. In different network sizes (N_V), **Fig. 5** shows the optimal service node ratio (N_S/N_V) with different value of the number of egress nodes (N_E). The results are obtained by solving Eq. (2)~(5). It is easy to find that the service node ratios are among range [0.05, 0.25] and increase with the parameter N_E. The reason is that the more egress nodes are, the more network nodes can be suitable for placing service.

Then, under different values of the parameter N_E, we illustrate the overall delay (calculated by Eq. (2)) of network traffics varying with the iteration rounds in **Fig. 6**.The results show that SLP-IQGA reaches rapidly the convergence state after about 100 iteration rounds and search towards the optimal solution by the quantum variation and quantum crossover (i.e. the step changes in the diagram). Due to the increase of the number of exports, network traffics can choose a more appropriate path for transmission, thus the overall transmission delay value of the network decreases with the increase of the parameter N_E.

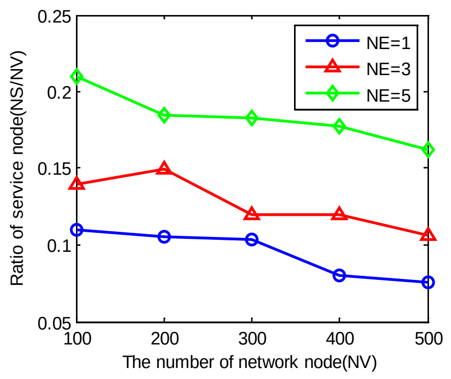


Fig. 5. Service node ratio in different network sizes

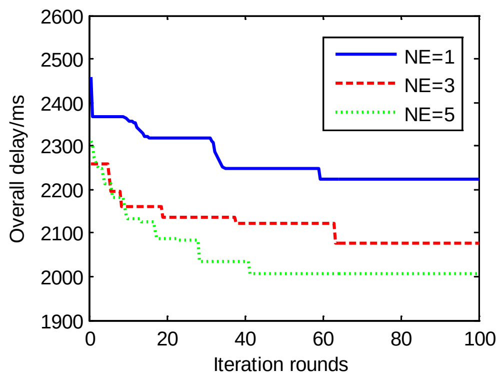


Fig. 6. Overall delay with iteration rounds

5.1.2 Comparison of different algorithms

We first compare our SLP-IQGA with the deployment strategy based on the genetic algorithm (denoted as SLP-GA) and the strategy based on the basic quantum genetic algorithm (denoted as SLP-QGA). Under the condition N_E=5, the simulation results of three strategies are shown in **Fig. 7**. The overall delays generated by various policies all increase with the expansion of network sizes while the delay of SLP-IQGA is lower than that of the other two strategies in each same network size. The reason is that IQGA policy has stronger search ability with some improving measures (e.g. dynamic rotation angle) within the solution space and can also search more suitable solutions.

In addition, according to the network topology with size N_V=100 and N_E=5, we execute SLP-IQGA to get the optimal deployment scheme and also produce a random scheme by random number generator (denoted as SLP-Random). We simulate 10 applications traffics that transport the network topology and compute the average delay of each traffic for two deployment schemes. In **Fig. 8**, we show the delay cumulative distribution function (CDF) for all the flows. For 50% of the flows, our SLP-IQGA can achieve 55ms delay while the random scheme needs 80ms. This is because the current widely used random strategy has difficulty carrying out a reasonable service deployment based on the actual situation of the network topology, but our IQGA can do it. Thus, the advantage of SLP-IQGA is significant and it is efficiently used to solve the service location problem.

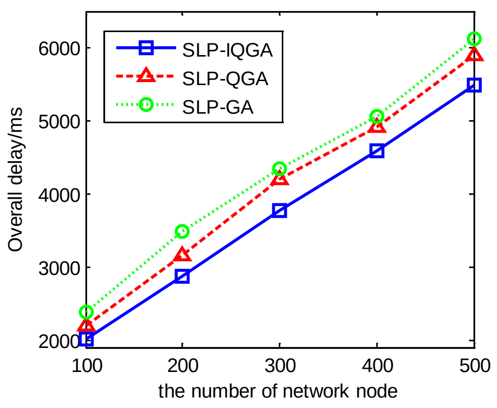


Fig. 7. Overall delay of each strategy under different network sizes

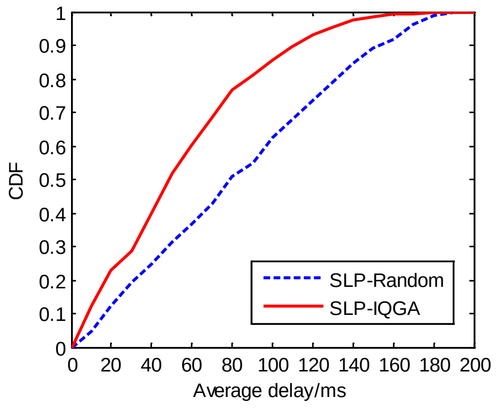


Fig. 8. Average delay distribution with the different deployment policies

5.2 Evaluation results of SAP-DCGA algorithm

In this section, we evaluate the performance of SAP-DCGA algorithm described in section 4.3. The GT-ITM tool is used to generate a 100-node topologies with the connection degree 10. Assuming that all nodes in the network topology have the same dominant resources, and the virtual middlebox instances of different type services have consistent resource demands, so we can ensure the load capacity Z at each node and set Z_j=1000 (j=1,…,100).

5.2.1 Satisfaction degree of requests

The main objective of SAP-DCGA algorithm is to satisfy the users' requirements as much as possible. So we define the measurement index, namely the satisfaction degree (*SD*), which can reflect the performance responding to the service requests and is calculated by

$$SD = \frac{1}{K \times J}\sum_{j=1}^{J}\sum_{k=1}^{K}\frac{n_{kj}}{R_{kj}}, \tag{31}$$

where R_{kj} is the requirement amount and n_{kj} is the actual placement amount for service s_k on the platform g_j. The greater the satisfaction degree is, the better the algorithm performs.

On the one hand, we analyze the scenario where there are sufficient resources on service platforms when the requests of service are few. **Fig. 9** shows the results of different number of service types (*K*) with the basic placement amount *B*=50. It is intuitive to find that the *SD* can rapidly converge to about 96%, and the value is also greater with the increase of the number of service types. This is because that the game process is more adequate when service types brings larger amount of game individuals in the game.

On the other hand, we discuss the scenario with resource storage caused by the total amount of requests being greater than the total load of platforms. In this case, **Fig. 10** plots the simulation results with the different number of service types (*K*) and different value (*B*), where the *SD* gradually decreases with the value *K* and under the same number of service types, the *SD* increases with the value *B* enhancement. Since the basic placement amount can guarantee the minimum placement amount of service, it is helpful to improve the *SD*. Therefore, the basic place amount (*B*) plays an important role in effectively distributing the network resource for different service requests.

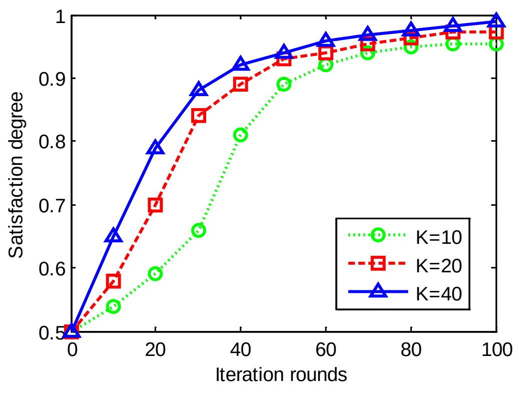


Fig. 9. Satisfaction degree of different number of service types

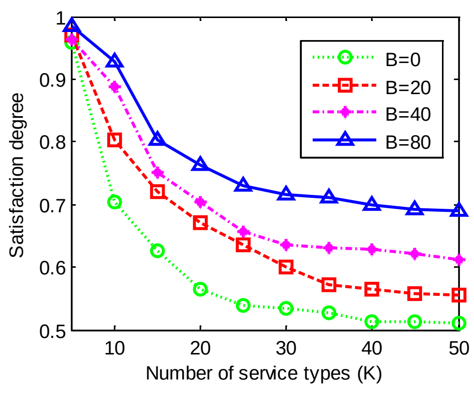


Fig. 10. Satisfaction degree with different basic placement amount

5.2.2 Fairness index of network resources

Furthermore, we use the parameter, i.e. Fairness Index (FI), to measure distributive justice of the SAP-DCGA algorithm. The fairness index is defined as follows:

$$FI = \frac{1}{J \times K}\sum_{j=1}^{J}\left[\left(\sum_{k=1}^{K} n_{kj}\right)^2 \Big/ \left(\sum_{k=1}^{K} n_{kj}^2\right)\right], \tag{32}$$

where the n_{kj} is the same to Eq. (31). The closer FI is to 1, the fairer the distribution result is.

In this simulation, we use the same experimental scenario with **Fig. 9** and **Fig. 10**. Under the condition of full resources, **Fig. 11** shows that with K=10, 20, 40, the values of FI can finally reach a stable state at about 0.95 and the lager the K value is, the better the results is. The results with resource shortage are shown in **Fig. 12**, where the fairness indexes display different changes with the increase of value B, and the FIs with B=20, 40 outperform that with B=0, 80. The reason is that the basic placement amount keeps the balance between the placement fairness index and satisfaction degree. Generally, with the different requirements of service, larger amount of basic placement can satisfy more requests, and restricts the increase of fairness. Hence, it is necessary to choose a proper value for the basic placement amount so as to obtain the tradeoff between satisfaction and fairness.

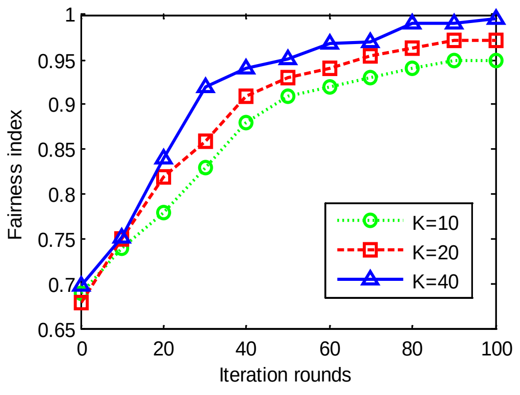


Fig. 11. Fairness index of different number of service types

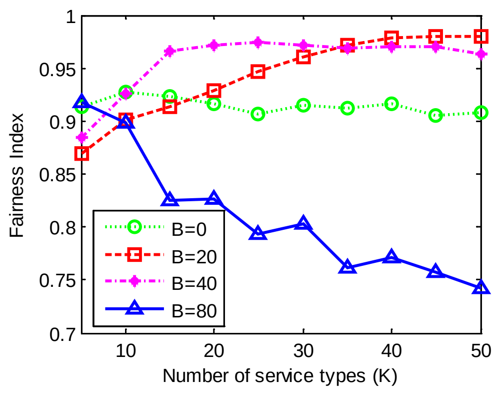


Fig. 12. Fairness index with different basic placement amount

5.2.3 Comparison of different algorithms

Furthermore, we compare SAP-DCGA with the other two typical placement strategies: (1) Proportional Policy [28] (denoted as SAP-PP), (2) Max-Min Fairness [29] (denoted as SAP-MMF). We set the number of service types K=20 and B=40 for our SAP-DCGA. When the node resources are relatively rich, we achieve the results shown in **Fig. 13** and **Fig. 14**.

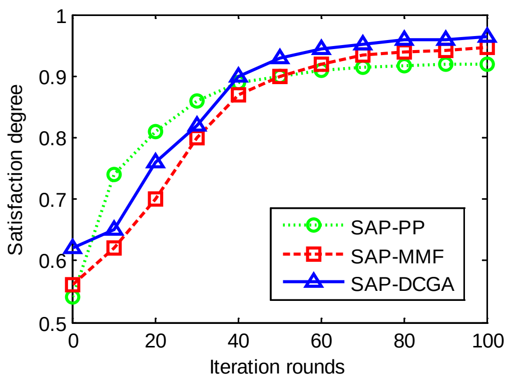


Fig. 13. Satisfaction degree of different strategies

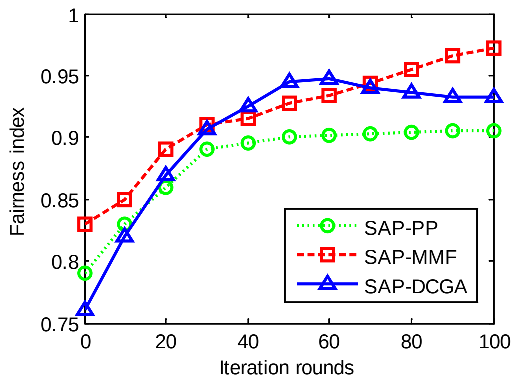


Fig. 14. Fairness index of different strategies

Fig. 13 shows the satisfaction degree responding to three strategies, where SAP-DCGA finally gets the highest satisfaction degree. **Fig. 14** demonstrates that the fairness of SAP-MMF is better than that of the other two strategies. At the aspect of the convergence speed, SAP-PP outperforms the other two strategies. The main reason is that SAP-PP approach distributes the resources according to the ratio of individual demand to total demand, which has a lower computational complexity. Due to the special design for the allocation fairness, SAP-MMF shows a better performance in fairness of resource distribution. In contrast, with the help of cooperative game model, SAP-DCGA has a better comprehensive advantage considering the aspects of the satisfaction, fairness and convergence.

5.3 Evaluation results of deployment mechanism

Based on the topology of **Fig. 2**, we try to verify the overall performance of the proposed optimization deployment mechanism (denoted as "Proposed"). In simulation, we assume that there are five types of virtual middlebox operating on general server 1 and server 2 and the load capacity of each server is Z=200. Using the method described in literature [27], we generate 100 simulation data flows, and the number of service requests of each flow is uniformly in the integer value from 1 to 5. Our method is compared with the naive method of the manual configuration (denoted as "Naive"), where the servers are placed on border nodes and the number of various types of services is evenly deployed on each server.

Fig. 15 shows the data traffic transmission delay of the two deployment mechanism (by "hop" units). The results illustrate that the traffic delay increases with the increase of the number of requests. And the larger the number of service requests is, the greater the value of the transmission delay can be reduced by our proposed method. Meanwhile, **Fig. 16** shows the satisfaction degree of service request of the two mechanism. With the increase in the number of requests, satisfaction degree gradually decreases. Compared with the naive method, the average transmission delay of the proposed method is reduced by 22.5%, while the request satisfaction is increased by 10.4%. It is demonstrated that the proposed method can optimize the service deployment by solving corresponding decision models and provides an effective solution to improve the service quality of virtual middleboxes and guarantee the network performance.

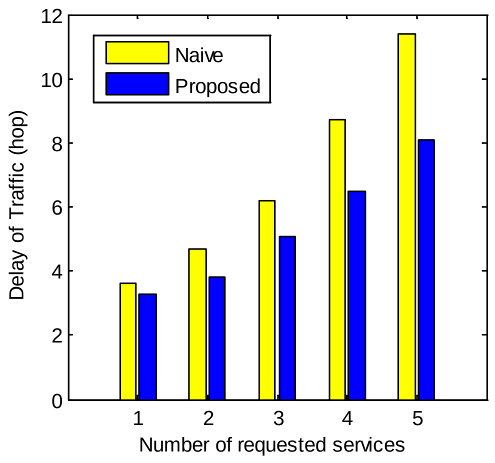


Fig. 15. Delay value of two mechanisms

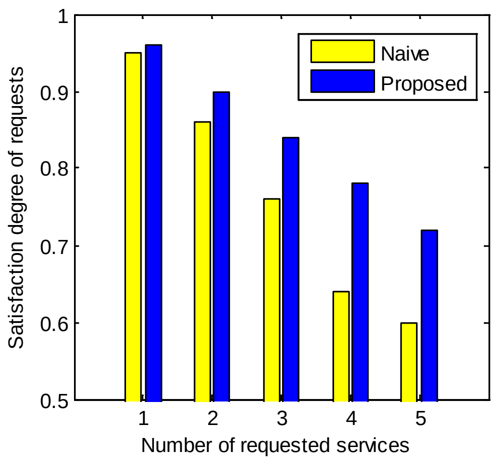


Fig. 16. Satisfaction degree of two mechanisms

6. Conclusion

According to service deployment problem in NFV and SDN network environment, we present an optimal deployment mechanism based on the improved quantum genetic algorithm and the cooperative game theory. First, through analyzing the demands of virtual middlebox service placement, we describe the service placement problem with respect to the additional transportation delay and limitation of network resources. Second, SLP-IQGA approach built on top of the improved quantum genetic algorithm is developed for the service location placement. For the service amount placement, we take an attempt towards using game-theoretic approaches to make the placement policy SAP-DCGA. Third, extensive simulations under experimental scenarios show that our proposed method can achieve better network benefits.

For future work, we will use our proposed approaches to help the construction of network function service chains, where more factors such as the link constraints will be considered. Additionally, the control models based on our proposed placement policy should be developed into a helpful tool for operators to plan service placement case in NFV or SDN networks.

References

[1] D. Joseph and I. Stoica, "Modeling middleboxes," *IEEE Network*, vol. 22, no. 5, pp.20-25, 2008. Article (CrossRef Link)
[2] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, V. Sekar, "Making middleboxes someone else's problem: network processing as a cloud service," in *Proc. of ACM SIGCOMM*, pp. 13-24, 2012. Article (CrossRef Link)
[3] V. Sekar, S. Ratnasamy, M. K. Reiter, N. Egi, and G. Shi, "The middlebox manifesto: enabling

innovation in middlebox deployment," in *Proc. of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, 2011. Article (CrossRef Link)
[4] M. Chiosi, D. Clarke, P. Willis, et al., "Network functions virtualisation –introductory white paper," *SDN and OpenFlow world congress*, Darmstadt, Germany, October 22-24, 2012. Article (CrossRef Link)
[5] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617 – 1634, 2014. Article (CrossRef Link)
[6] I. Banerjee, I. Perera, A. Nag, J. Choudhury, "Incorporating Smart Software-defined Networks to Enhance Resilience and Survivability in the Cloud: Survey of Current Challenges and Future Research Scopes," *Smart Computing Review*, vol. 5, no. 4, pp. 308-316, Aug. 2015. Article (CrossRef Link)
[7] S. Liu, W. Jia. "An adaptive virtual machine location selection mechanism in distributed cloud," *KSII Transactions on Internet and Information Systems*, vol. 9, no. 12, pp. 4776-4798, Dec. 2015. Article (CrossRef Link)
[8] V. Sekar, N. Egi, S. Ratnasamy, M. Reiter, and G. Shi, "Design and implementation of a consolidated middlebox architecture," in *Proc. of the 9th USENIX conference on Networked Systems Design and Implementation*, NSDI'12, pp. 1–17, Berkeley, USENIX Association, 2012. Article (CrossRef Link)
[9] A. Gember, R. Grandl, A. Anand, T. Benson, and A. Akella, "Stratos: virtual middleboxes as first-class entities," *Technical Report TR1771*, University of Wisconsin-Madison, 2012. Article (CrossRef Link)
[10] A. Gember, P. Prabhu, Z. Ghadiyali, and A. Akella, "Towards software defined middlebox networking," in *Proc. of the 11th ACM Workshop on Hot Topics in Networks*, HotNets-XI, 2012. Article (CrossRef Link)
[11] A. Gember-Jacobson, R. Viswanathan, C. Prakash, et al., "OpenNF: enabling innovation in network function control," in *Proc. of the ACM SIGCOMM'14*, Chicago, IL, USA, 2014. Article (CrossRef Link)
[12] Z. A. Qazi, C. C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," in *Proc. of the ACM SIGCOMM'13*, Hong Kong, China, pp. 27-38, August 12-16, 2013. Article (CrossRef Link)
[13] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM CCR*, vol. 38, no.2, 2008. Article (CrossRef Link)
[14] Y. Zhang, N. Beheshti, L. Beliveau, G. Lefebvret, R. Manghirmalani, R. Mishra, "StEERING: a software-defined networking for inline service chaining," in *Proc. of the 21st IEEE International Conference on Network Protocols (ICNP)*, pp.1-10, 7-10 Oct., 2013. Article (CrossRef Link)
[15] S. K. Fayazbakhsh, L. Chaing, V. Sekar, M. Yu, and J. C. Mogul, "Enforcing network-wide policies in the presence of dynamic middlebox actions using FlowTags," in *Proc. of NSDI*, 2014. Article (CrossRef Link)
[16] A. Gushchin, A. Walid and A. Tang, "Scalable routing in SDN-enabled networks with consolidated middleboxes," in *Proc. of HotMiddlebox'15*, London, United Kingdom, August 17-21, 2015. Article (CrossRef Link)
[17] G. Z. Cheng, H. C. Chen, Z. M. Wang, P. Yi, F.Y. Zhang, and H. C. Hu, "Towards adaptive network nodes via service chain construction," *IEEE Transactions on network and service management*, vol. 12, no. 2, pp.248-262, June, 2015. Article (CrossRef Link)
[18] Z. Guo, M. Su, Y. Xu, Z. Duan, L. Wang, S. Hui, H. Chao. "Improving the performance of load balancing in software-defined networks through load variance-based synchronization," *Computer Networks*, vol. 68, no. 11, pp. 95-109, 2014. Article (CrossRef Link)
[19] A. Basta, W. Kellerer, M. Hoffmann, H. Morper, and K. Hoffmann, "Applying NFV and SDN to

LTE mobile core gateways; the functions placement problem," in *Proc. of the 4th workshop on all things cellular: operations, applications and challenges*, pp. 33-38, 2014. Article (CrossRef Link)

[20] A. Baumgartner, V. S. Reddy, T. Bauschert, "Mobile core network virtualization: A model for combined virtual core network function placement and topology optimization," in *Proc. of 1st IEEE Conference on Network Softwarization (NetSoft)*, London, pp. 1-9, April 13-17, 2015. Article (CrossRef Link)

[21] C Hu, J Yang, H Zhao, J. Lu "Design of all programable innovation platform for software defined networking," in *Proc. of Presented as part of the Open Networking Summit* 2014, Santa Clara, CA, 2014. Article (CrossRef Link)

[22] Z Guo, Y Xu, M Cello, J Zhang，Z Wang, M. Liu, H. Chao, "JumpFlow: reducing flow table usage in software-defined networks," *Computer Networks*, vol. 92, no. 3, pp. 300-315, 2015. Article (CrossRef Link)

[23] A. Malossini, E. Blanzieri, T. Calarco, "Quantum genetic optimization," *IEEE Trans. on Evolutionary Computation*, vol. 12, no. 2, pp. 231-241, 2008. Article (CrossRef Link)

[24] J. Guo, F. M. Liu, D. Zeng, J. C.S. Lui, H. Jin, "A cooperative game based allocation for sharing data center networks," in *Proc. of INFOCOM*, 2013. Article (CrossRef Link)

[25] H. Yaiche, R. Mazumdar, and C. Rosenberg, "A game theoretic framework for bandwidth allocation and pricing in broadband networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 8, no. 5, pp. 667– 678, 2000. Article (CrossRef Link)

[26] E. Egura, K. Calvert, S. Bhattacharjee, "How to model an internetwork," in *Proc. of IEEE International Conference on Computer Communications*, San Francisco, pp. 594 – 602, 1996. Article (CrossRef Link)

[27] Gebert S, Pries R, Schlosser D, et al., "Internet access traffic measurement and analysis," in *Proc. of the 4th International Conference on Traffic Monitoring and Analysis*, Berlin, volume 7189 of LNCS, pp. 29–42, 2012. Article (CrossRef Link)

[28] A. Ghosh, S. Ha, E. Crabbe, J. Rexford, "Scalable multi-Class traffic management in data center backbone networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 12, pp.2673-2684, 2013. Article (CrossRef Link)

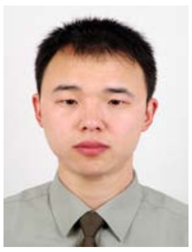
[29] E. Danna, A. Hassidim, H. Kaplan, et al., "Upward max min fairness," in *Prof. of INFOCOM*, pp.837-845, 2012.Article (CrossRef Link)

Gang XIONG received his B.E. and M.E. degrees in 2009 and 2012, respectively. He is currently a Ph.D. candidate in National Digital Switching System Engineering and Technological R&D Center (NDSC). His research interests are future network and network security. Email: xg1226@126.com

Penghao SUN received his B.E. degrees in 2014. He is currently pursuing a M.E. degree in NDSC. His research interests are future network and network data plane.

Yuxiang HU is an assistant professor in National Digital Switching System Engineering and Technological R&D Center (NDSC), China. His research interests mainly include network security, routing protocols and future network.

Julong LAN is the full Professor in National Digital Switching System Engineering and Technological R&D Center (NDSC), China. His research interests mainly include network communication, future network architecture, and network security.



Kan Li received the M.E. degree in 2009 and is currently an assistant with department of information science. His research interests include information security and future network.