2016 IEEE 22nd International Conference on Parallel and Distributed Systems

Optimized Virtual Network Functions Migration for NFV

Jing Xia, Zhiping Cai, Ming Xu
College of Computer
National University of Defense Technology
Changsha, 410073, China
{jingxia, zpcai, xuming}@nudt.edu.cn

***Abstract*—Combining with software-defined networking and IT virtualization technologies, Network Function Virtualization (NFV) has been proposed as an important technology to speed up deployment of new network services. VNF (Virtual Network Function) migration is a critical step to redeploy virtual network functions for providing better network services. Previous work in virtual network function migration primarily focused on the migration mechanism, including maintaining internal state consistency and reducing migrating time. In this paper, we address the problem of optimally migrating virtual network functions. As the computing and network resource requirement of virtual network functions have been changed, these virtual network functions have to be migrated to meet the computing and network resource constraints. As the SDN controllers conducted technology is used to migrate the virtual network functions, the migration cost depends on the buffer size of controllers and the time of transferring the internal state of virtual network functions. A cost model is proposed to evaluate the migration cost. The problem of optimal virtual network functions migration with satisfying computing and network resource constraints is NP-hard. A heuristic algorithm is proposed for computing the approximate solution. The effectiveness of the algorithms is validated by simulations evaluation.**

***Index Terms*—NFV, SDN, virtual Network Function, Migration**

I. Introduction

In NFV-featured network, Virtual Network Functions (VNFs) are network traffic processing softwares which run on one or more virtual machines to handle specific network functions on top of the hardware networking infrastructure (routers, switches, etc). Individual virtual network functions can be connected or combined together as building blocks to offer a full-scale networking communication service. Combined with Software-Defined Networking (SDN), VNFs can be easily deployed or redeployed at any location of a network [1].

A major challenge in NFV-featured dynamic network is how to maintain reasonable VNF deployment to adapt to the changes in the network. The agile deployment mechanism of VNF can be used to address this challenge. When a network changes, administrator can remove the obsolete VNF and redeploy a new one at a better location. However, many network function applications, like firewall or IDS (Intrusion Detection System), need to track the operating state and characteristics of flows traversing through it, which means the history information must be kept in these network functions. In other words, the network function is stateful, and the redeployment of the stateful network function will break the continuity of the flow state tracking. Thus, the internal state must be transferred along with network function migration.

Existing works related to the VNF deployment and migration usually focus on proposing new deployment strategies [2] and migration mechanisms [3] [4]. But the migration cost didn't be considered in those studies. In fact, the migration cost is the key factor in the migration process. It will affect the decision that how to choose candidate VNFs to be migrated and find out the suitable migration target position.

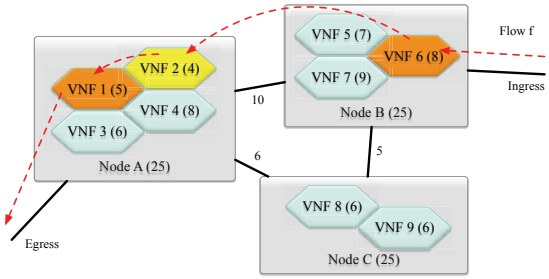
We show an example of VNF migration in Fig.1. In this scenario, 9 VNFs run on 3 network nodes. A network flow travels from the ingress to the egress. The numbers over the links represent link delay and the numbers shown in bracket with the node name are the computing resources available at the nodes, the numbers in bracket beside VNF name are the computing resources consumption by the VNF. All of computing resource in each node is 25 units.

Fig. 1(a) shows the original status before VNF migration occurs. The network flow f travels through the network and is served by three different VNFs with the exact order: VNF6, VNF2 and VNF1. In the original state, VNF6 is placed on the Node B, both VNF1 and VNF2 are placed on the Node A. Their computing resource consumption are 8, 5 and 4 units, respectively. As shown in Fig. 1(a), all VNFs can work well.

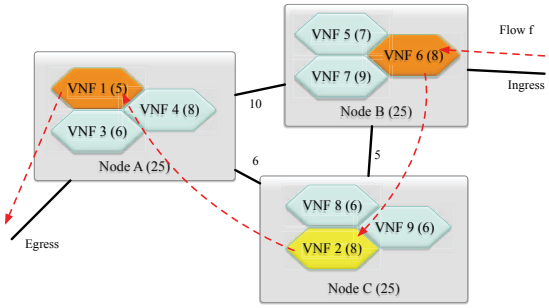
In some cases, the resource requirement of VNFs will be changed. For example, the firewall needs to install more filter rules. Assuming the VNF2 needs to raise up its computing resource requirement to 8 units. Then the total computing resource consumption in Node A will raise up to 27 units, which is greater than 25 and beyond the computing resource that Node A could provide. The administer have to redeploy or migrate the VNFs in node A for guaranteeing the service quality. Fig. 1(b) and (c) illustrate two possible migration plan.

The plan I migrates VNF2 to Node C, which reduces computing resource consumption in Node A to 19 units, increases computing resource consumption in Node C to 20 units, and brings 11 units delay to flow f. The plan II migrates VNF5 to Node C, which reduces computing resource consumption in Node A to 22 units, increases computing resource consumption in Node C to 17 units, and brings 22 units delay to flow f. Hence, different strategies will produce

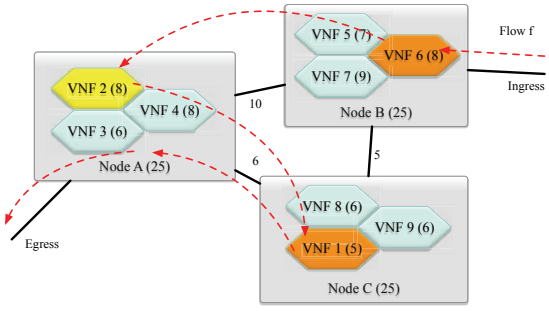
1521-9097/16 $31.00 © 2016 IEEE
DOI 10.1109/ICPADS.2016.51



(a) Original NFs placement and flow route



(b) NFs migration plan I



(c) NFs migration plan II

Fig. 1. Virtual NF placement and two migration plans

different workload for different node, different network delay for different flow.

For solving the problem of resource shortness in the nodes with a number of VNFs, we could use the deployment schema to redeploy and migrate all VNFs. But existing deployment strategies didn't consider the migration cost. In fact, an optimized deployment schema may be not optimal in the migration scenario. For example, in the migration scenario shown in Fig. 1. If the minimum flow delay is the optimal goal of the deployment strategy, migration plan I can be significantly improved in flow delay against plan II. However, if the migration of VNF2 would bring a huge cost to the migration task, and 22 units delay in flow f is tolerable, the migration plan II will be better than plan I. Apparently, *migration cost* should be considered carefully in VNF migration cases.

To address this problem, we propose a cost model to measure the VNF migration cost. The formulation of the optimal VNF migration problem with resources constraints for dynamic networks is presented, and the integer programming approach is provided. As the optimal problem is NP-hard, we develop a heuristic algorithm to find out an approximate optimal migration plan for typical dynamic scenario and evaluate its performance using simulated network.

The rest of the paper is organized as follows: related works are discussed in Section II. A VNF migration cost model for dynamic network is proposed in Section III. Based on the cost model, we describe the formulation of the optimal VNF migration problem in section IV. In the next section, we propose a heuristic algorithm to solve the optimization problem. The effectiveness of the algorithm is verified by simulations evaluations in Section VI. Finally, conclusions are given in the last section.

II. Related Work

There are a number of studies to address the implementation mechanism of VNF deployment and migration. Sekar et al. innovated middlebox deployment with the software-centric middlebox implementations running on general-purpose hardware platforms managed via open and extensible management APIs [5]. Gember et al. realized a software-defined middlebox networking framework to simplify the management of complex and diverse functionalities [6]. In the scenarios of NFV and SDN, Gember et al. designed a control plane called OpenNF [3], which could provide efficient, coordinated control of both internal middlebox state and network forwarding state.

The service chain is the key to connect the individual virtual network functions to offer efficient network service. Cheng et al. used the simulating annealing algorithm for the combinational problem of service chains, which could manage network services in an efficient and scalable way [7]. Xiong et al. designed a quantum genetic algorithm to resolve the service placement and minimize the network transport delay [2]. Ye et al. proposed a algorithm to address the JTDM (Joint Topology Design and Mapping of Service Function Chains) problem, and proposed a scalable and reliable strategy which can significantly reduce the network reconfigurations and enhance the service reliability [8].

Another related area is virtual network embedding, which means efficiently map the virtual nodes and virtual links onto the substrate network resource. A number of studies consider virtual network embedding with the resource constraints, for example the computing and network resource constraints [9] or the security constraint [10]. Cai et al. also addressed the virtual network embedding problem for evolving network [11]. But the virtual network functions are different with the virtual network on some characteristics and resource requirements. For example, the VNFs redeploy task needs to keep the internal state. Hence, those algorithms cannot be used to solve the VNF migration problem directly.

Currently, a few of virtual network function placement strategies are proposed to deploy or redeploy the VNFs. The network resource constraint [12], the elasticity [13] and flexibility [14] of the placement plan are discussed. But these

approaches do not address the issue of migration cost. We propose a migration cost model and a heuristic algorithm to decrease the migration cost.

III. VNF Migration Cost

A number of VNF migration mechanisms have been proposed to reduce the migration time and hold the internal state. The migration cost didn't be considered in those studies. For example, the OpenNF [3] proposed a migration mechanism called *loss-free move* to migrate internal VNF state. One pair of northbound and southbound interface protocol and a two-stage algorithm have been proposed to control the VNF migration process. To fulfill the migration, the SDN controller extracts internal state from the source VNF and writes them into the target VNF. During the migration process, the source VNF has to send all received packets to controller until the internal state migration has finished. The SDN controller must store all packets in its main memory, which is the significant bottleneck of the migration.

Assuming a VNF migration affects a network flows whose traffic rates are b packets per second (pps), and the state migration operations take s seconds per flow, then the controller must buffer up to $s \times a \times b$ packets. Even equipped with optimization mechanism, e.g., late-locking and early-release (LLER), OpenNF still need 2K packets buffer size for transferring internal state of 800 flows while the traffic rate for these flows is just 20K pps (0.15Gbps) in a very short time (between two Bro IDS instances $s \approx 8$ ms) [15].

Due to the limitation of the main memory size, the buffer is the bottleneck of VNF migrations. The buffer may be overflowed when the migration time is too long or the number of active flows is too large. In comparison with the transferred aggregated traffic, the size of transferred VNF internal state is relatively small. Hence, for the migration process, the migration overhead can be defined as the size of aggregated traffic which are transferred to the controller.

Therefore, the cost to migrate one VNF is the total traffic rate it served. For a VNF which serves a network flows, $f_1, f_2, f_3, ...f_a$, whose traffic rates are respectively $b_1, b_2, b_3, ...b_a$ packets per second,, the cost C_m of migrating this VNF can be calculated by the following equation:

$$C_m = \sum_{i=1}^{a} b_i \tag{1}$$

IV. System Model and Problem Formulation

A. Physical Network

We use one undirected graph, $G = (N, E)$, to represent the physical network, where N is the set of its nodes and E is the set of its edges. Each node corresponds to a OpenFlow switch, and each edge is a link connecting two switches. It will be assumed for simplicity that every physical machine is connected with a single switch. Assuming there are k nodes and h edges in the network, $N = \{n_i | 0 < i \leq k\}$ and $E = \{e_i | 0 < i \leq h\}$.

Assuming there are l network flows in the network, and we denote them as set F, and ith flow denoted as $f_i \in F$, whose flow traffic rate is b_i.

Service function chain is used to describe the VNFs need to serve the flow. We denote it as tuple S_i, which has p_i VNF elements, denoted as $S_i = (v^0_{s_i}, v^1_{s_i}, v^2_{s_i}, ..., v^{p_i}_{s_i}, v^{p_i+1}_{s_i})$, where $v^j_{s_i}$ means the jth VNF in the service function chain, and $v^0_{s_i}$, $v^{p_i+1}_{s_i}$ means the source and the target of the flow. All the VNFs must be contained in at least one of service function chains. We denote the set of VNF as V, and ith VNF as $v_i \in V$. The total number of VNF in the network is m.

Each VNF needs to be placed on one network node. We denote this relationship as function R, $n_j = R(v_i)$ means VNF v_i is placed on node n_j.

B. Computing Resource and Delay Constraints

In this paper, we consider computing resource for node constraint and link delay for network resource constraint.

The computing resource include the CPU and memory capacity. The node-capacity function $P(n)$ is designed to represent the capacity of computing resource in the node n and the VNF requirement function $Q(v)$ is proposed to represent the resource requirement of the VNF v.

Before the migration, the NF computing resource consumption is $Q_o(v_i)$, and after the migration, the expected computing resource consumption is $Q_t(v_i)$.

The link delay constraint is also considered in the model. Actually, link delay represents the bandwidth constraint and capacity for each link. The edge-delay function $D(e)$ assigns a non-negative weight to each link in the network. The value $D(e)$ associated with edge is a estimation value of total delay that packets travels on the link. It is assumed all paths in networks are loop-free paths. Let the set $E'(Path(u, w))$ represent the set of edges which are in the shortest path from node u to node w, so we have

$$Delay(Path(u, w)) = \sum_{e \in E'(Path(u,w))} D(e) \tag{2}$$

As every flow has a delay constraint, we denote the delay constraint of flow f_i as delay constrain function $L(f_i)$.

C. Optimal Objectives

When the workload or the resource have been changed, a number of VNFs have to be migrated. We take the migration problem as an optimization problem that aims at minimizing the total migration cost under the constraint of delay and computing resource.

We define the binary variable $x_{ij} \in \{0, 1\}$ to indicate whether the VNF v_i is located the node n_j after the migration. The indicate variable $x_{ij} = 0$ means that VNF v_i is not placed on node n_j after the migration; otherwise, v_i is the placed on node n_j after the migration.

$$x_{ij} = \begin{cases} 0 (n_j \neq R(v_i)) \\ 1 (n_j = R(v_i)) \end{cases} \tag{3}$$

We introduce binary variable y_{ij} to indicate network status before the migration. It is similar to the variables x_{ij}, $y_{ij} = 0$ means that VNF v_i is not placed on node n_j before the migration; otherwise, v_i is the placed on node n_j before the migration.

$$y_{ij} = \begin{cases} 0(n_j \neq R(v_i)) \\ 1(n_j = R(v_i)) \end{cases} \tag{4}$$

Therefore, we can use the indicate variable I_i to indicate whether the VNF v_i has been migrated in the current migration process.

$$I_i = \sum_{j=1}^{k} x_{ij} y_{ij} \tag{5}$$

When $I_i = 0$ indicates the VNF v_i has been migrated in the current migration process, and $I_i = 1$ indicates the VNF v_i has not been migrated.

Variable z_{ij} is used to indicate the relation between VNFs and flows, $z_{ij} = 1$ means that VNF v_i belongs to the S_j, which is the service function chain of flow f_j; otherwise, v_i is not one part of the service function chain S_j. Denote the element set of S_j as $M(S_j)$,so we have:

$$z_{ij} = \begin{cases} 0(n_i \notin M(S_j)) \\ 1(n_i \in M(S_j)) \end{cases} \tag{6}$$

As we discussed in section III, the cost of migrating a VNF equals to the aggregate transferring traffic rate during the whole migration process. Hence we can formally define the cost function as:

$$C_M = \sum_{i=1}^{m} \sum_{j=1}^{l} z_{ij} b_j \times (1 - I_i) \tag{7}$$

D. Integer Programming Formulation

The optimal problem of virtual network function migration for dynamic network can be stated as follows: Given a network G, flow set F, original VNF placement schema $Y = \{y_{ij} | v_i \in V \& n_j \in N\}$, and the computing resource request of VNFs, determine a VNF migration scheme $X = \{x_{ij} | v_i \in V \& n_j \in N\}$ in G such that:

1)the cost of migration task is minimum;

2)all VNFs in V are placed on one node;

3)the computing resource request of all VNF should be satisfied;

4)the delay on each flow should be less than the maximum delay bound $L(f)$.

The optimal problem can be considered as an integer linear programming (ILP) problem expressed as follows.

The objective is:

$$MinimizeC_M \tag{8}$$

and the subjects are below:

TABLE I
MAIN NOTATIONS

Notation	Description
$G = (N, E)$	Physical network
$F = (f_i \| 0 < i \leq l)$	Network flow set
$V = (v_i \| 0 < i \leq m)$	VNF set
S_i	Service function chain of the flow i
b_i	Traffic rate of the flow i
k	Number of the nodes in network
m	Number of the VNFs in network
l	Number of the flows in network
x_{ij}	Whether VNF v_i is on the network node n_j after the migrate
y_{ij}	Whether VNF v_i is on the network node n_j before the migrate ocurred
z_{ij}	Whether VNF v_i belongs to the service chain S_j
I_i	Whether VNF v_i is migrated
$R(v_i)$	Location of VNF v_i
$Q(v_i)$	Computing resource consumption of VNF v_i
$P(n_i)$	Computing resource capacity of node v_i
$Path(v, w)$	Shortest path between node u and w
$D(e)$	Delay of the link e
$Delay(Path(u, w))$	Delay of a path between u and w
$L(f_i)$	Delay constraint of a flow i

$$\sum_{j=1}^{k} x_{ij} = 1(\forall v_i \in V) \tag{9}$$

$$\sum_{i=1}^{m} x_{ij} Q_t(v_i) \leq P(n_j)(\forall n_j \in N) \tag{10}$$

$$\sum_{q=1}^{p_i} Delay(Path(R(v_{s_i}^q), R(n_{s_i}^{q+1})) \leq L(f_i)(\forall f_i \in F) \tag{11}$$

$$x_{ij} \in \{0, 1\}(\forall v_i \in V, n_j \in N) \tag{12}$$

The constraint (9) ensures that each VNF is mapped to one node in G exactly. The constraint (10) ensures that located physic node can satisfy the VNF computing resource requirement. The constraint (11) ensures that total delay of the path which the flow travels through does not exceed the delay constraint.

Main notations are summarized in Table I.

E. Computational Complexity Issues

The optimal problem of VNFs migration is obviously NP-hard problem. Thus we do not have possibility to get an exact polynomial time algorithm for this problem.

V. HEURISTIC ALGORITHM

As the optimal migration problem with resource constraints is NP-hard, we examine polynomial time heuristics in this section. The objective of this heuristic is to minimize the

cost function, which is the total migration cost of one VNF migration task.

The heuristic algorithm presented in this section is to minimize the cost function C_M, while satisfying computing resource constraint and delay constraint.

The heuristic algorithm consists of three steps. In the first step, our proposed algorithm finds out all the computing resources depleted node as the migration source set. In the second step, for each node in the migration source set, the algorithm calculates out the candidate target set for each VNF being placed on it, i.e. the set of network nodes satisfying the delay constraint for that VNF. In the third step, the algorithm locate the VNF migration target which must be migrated by using greedy strategy. For each node in the migration source set, the heuristic algorithm repeatedly greedily picks the VNF which has the smallest migration cost to perform migration, until the source node has available computing resource. For migration target, algorithm greedily picks a network node having enough computing resource from candidate target set to satisfy the VNF computing resource constraint, while the node has maximum available computing resource in the candidate target set.

A. Compute Migration Source Set

The first step of our heuristic algorithm acquire the migration source set for migration task. Normally, a migration task is triggered by lacking of computing resource in network node. The computing resource consumption by VNFs is dynamic with service requirement changing. The controller can collect computing resource consumption and the upcoming request of all the VNFs. We define computing resource load ratio of a network node as the ratio of total upcoming computing resource consumption on the node computing resource capacity. Whether a network node is labeled as computing resources depleted is determined by the computing resource load ratio which is greater than the threshold ρ or not.

The proposed algorithm searches all nodes in network and adds nodes, which the sum of the computing resource consumption of all the VNFs placed on it is greater than ρ, into the migration source set. The detail of computing migration source set algorithm are shown in Algorithm 1.

B. Compute Candidate Target Set

In this step, our heuristic algorithm acquire the candidate target set for each VNF in all the nodes belong to migration source set. The formal description of candidate target set of related VNF is below: For a VNF v, denote the flow which it serves as set F^v. for each flow $f_i^v \in F^v$, the candidate target set of VNF n, denoted as $T(v) \subseteq N$, with satisfying that: for any node $u \in T(v)$, for all the flow $f_i^v \in F^v$, satisfy the delay constraint by replace v's location to u.

For any node in network, we can use Breadth-First method to search maximum candidate target set of each VNF in all the nodes belong to migration source set. The detail of candidate target set algorithm are shown in Algorithm 2.

Algorithm 1 Computing Migration Source Set Algorithm

```
Migration Source Set C' ← ∅
for i ← 1 to |N| do
    sum = 0
    for j ← 1 to |V| do
        if v_j is placed on n_i then
            sum ← Q_t(v_j)
        end if
    end for
    if sum/P(n_i) ≥ ρ then
        C' ← C' ⋃ {n_i}
    end if
end for
return C'
```

Algorithm 2 Computing Candidate Target Set Algorithm

```
Candidate Target Set T(v) ← ∅
for i ← 1 to |N| do
    flag ← True
    for j ← 1 to |F^v| do
        delay' ← delay of the flow f_j when v's location changed to n_i
        if delay' > L(f_j) then
            flag ← False
            break
        end if
    end for
    if flag = True then
        T(v) ← T(v) ⋃ {n_i}
    end if
end for
return T(v)
```

C. Migrate virtual nodes

Some VNFs placed on the node which in the migration source set must be migrated to solve the computing resource lacking. Our algorithm firstly sorts the migration source node with decrease by the computing resource load ratio, then calculate the VNF migration plan for the first node. For a migration source node, the algorithm firstly sorts VNFs placed on this node with increase by the migration cost, then try to migrate the VNFs off the node from the first VNF until the computing resource load ratio be less than ρ.

For migrating one VNF, the algorithm picks the candidate target node with the minimal computing resource load ratio from its candidate target set. If the picked candidate target node cannot satisfy the node resource requirement, the algorithm will skip this VNF, then try to migrate another one in migration source node. The detail of migration algorithm is shown in Algorithm 3. Function $F(g)$ in Algorithm 3 represents the available computing resource on network node g.

Algorithm 3 Migrating Algorithm

sorts migration source node with decrease by the computing resource load ratio
for $i \leftarrow 1$ **to** $|C'|$ **do**
  sorts VNF set V^{n_i} placed on node n_i with increase by the migration cost
  $Q \leftarrow \sum_{j=1}^{|V^{n_i}|} Q_t(v_j)$
  for $j \leftarrow 1$ **to** $|V^{n_i}|$ **do**
    pick the network node g in $T(v_j)$ with the maximum available compting resource
    $r \leftarrow F(g) - Q_t(v_j)$
    if $r > 0$ **then**
      $Q \leftarrow Q - Q_t(v_j)$
      if $Q/L(n_i) < \rho$ **then**
        $x_{ji} \leftarrow 1$
        break
      end if
    end if
  end for
end for
return X

VI. Simulations

In this section, we evaluate the performance of the proposed heuristic algorithm on several different topologies and parameter settings.

A. Simulation Setup

We generate the network topologies by using the GT-ITM tool [16], The network scenario initially has 100 nodes and around 500 links, a scale that corresponds to a medium sized ISP. The computing resources of the network nodes are real number uniformly distributed between 30 and 100. The link delay follows a uniform distribution from 10 to 100 units. The number of flow in the network are random and the number of VNFs for one flow's service function chain is randomly determined by a uniform distribution between 2 and 10. The traffic rate of flows are following a uniform distribution between 100K bits per second (bps) and 1M bps. Computing resource consumption reported by VNFs are randomly determined by a uniform distribution between 3 and 20.

The migration task is triggered by altering VNF's computing resource consumption. Alternations are generated by the following rule: randomly pick certain number of VNFs as the candidate set A, and raise the computing resource consumption of each VNF in set A by a uniform distribution between 30% and 70%.

B. Scalable Evaluation

Fig. 2 shows the maximal memory consumption by package buffer in controller for one VNF migration task as a function of the stepwise adding random 25 nodes to network, while initial flow number is 20, and the VNF number in set A is fixed 50. The curves represent the instances with different flow numbers, respectively. It is shown the memory consumption size is slowly increased with flow density increase, and slight increase with network scales up. It is due to the increase of flow density leads increase aggregate traffic rate of VNFs, and a slight increase of the migration time by increase the number of network nodes. But the VNF density is relatively low, which makes the influence is small for migration cost as the flow density grows when the number of migrated VNF is a constant.

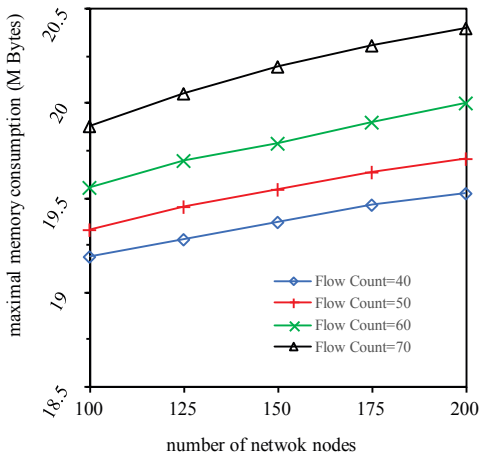


Fig. 2. Performance evaluation for scalable

C. Performance Evaluation

We compare the performance of our greedy algorithms with the optimal solution and the random algorithms. The optimal results are obtained by using integer programming approach. The random algorithms are using random picking strategy for migrating VNF.

Fig. 3 shows the performance comparison of algorithms, the bars represent the instances with different algorithm. We could obtain similar results under other parameter settings. Our results show that our heuristic algorithm is better than the random algorithm and the cost of our algorithm is close to the optimal results.

VII. Conclusion

In this paper, we address the optimal problem of VNFs migration for dynamic networks. We propose a cost model to evaluate the migration cost. We formulated the optimal problem as an integer programming problem, which is proved as NP-hard. A heuristic algorithm is proposed and validated by simulations evaluation.

Acknowledgments

The authors would like to thank the National Natural Science Foundation of China under Grant Nos. 61379144, 61379145, 61363071, 61402275, 61501482 for their supports of this work.

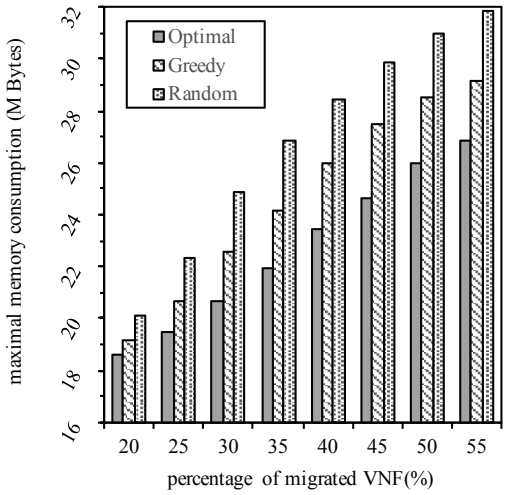


Fig. 3. Migration Cost Comparisons of Algorithms

REFERENCES

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

[2] G. XIONG, Y.-x. HU, L. TIAN, J.-l. LAN, J.-f. LI, and Q. ZHOU, "A virtual service placement approach based on improved quantum genetic algorithm," *Frontiers of Information Technology & Electronic Engineering*, vol. 17, no. 7, pp. 661–671, 2016.

[3] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "OpenNF: Enabling Innovation in Network Function Control," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM '14. New York, NY, USA: ACM, 2014, pp. 163–174. [Online]. Available: http://doi.acm.org/10.1145/2619239.2626313

[4] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Wareld, "Split/Merge: System Support for Elastic Execution in Virtual Middleboxes," in *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, ser. NSDI '13, 2013, pp. 227–240. [Online]. Available: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/rajagopalan

[5] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, "Design and Implementation of a Consolidated Middlebox Architecture," in *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, ser. NSDI '12, 2012, pp. 323–336. [Online]. Available: https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/sekar

[6] A. Gember, P. Prabhu, Z. Ghadiyali, and A. Akella, "Toward Software-defined Middlebox Networking," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XI. New York, NY, USA: ACM, 2012, pp. 7–12. [Online]. Available: http://doi.acm.org/10.1145/2390231.2390233

[7] G. Cheng, H. Chen, H. Hu, Z. Wang, and J. Lan, "Enabling network function combination via service chain instantiation," *Computer Networks*, vol. 92, Part 2, pp. 396–407, Dec. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128615003254

[8] Z. Ye, X. Cao, J. Wang, H. Yu, and C. Qiao, "Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization," *IEEE Network*, vol. 30, no. 3, pp. 81–87, May 2016.

[9] M. T. B. H. d. M. X. H. Andreas Fischer, Juan Felipe Botero, "Virtual network embedding: A survey," in *IEEE Communications Surveys and Tutorials 15(4)*, Dec 2013, pp. 1888–1906.

[10] S. Liu, Z. Cai, H. Xu, and M. Xu, "Security-aware virtual network embedding," in *2014 IEEE International Conference on Communications (ICC)*, June 2014, pp. 834–840.

[11] Z. Cai, F. Liu, N. Xiao, Q. Liu, and Z. Wang, "Virtual network embedding for evolving networks," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, Dec 2010, pp. 1–5.

[12] F. Wang, R. Ling, J. Zhu, and D. Li, "Bandwidth guaranteed virtual network function placement and scaling in datacenter networks," in *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*, Dec 2015, pp. 1–8.

[13] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic virtual network function placement."

[14] K. Kawashima, T. Otoshi, Y. Ohsita, and M. Murata, "Dynamic placement of virtual network functions based on model predictive control," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, April 2016, pp. 1037–1042.

[15] A. Gember-Jacobson and A. Akella, "Improving the Safety, Scalability, and Efficiency of Network Function State Transfers," in *Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, ser. HotMiddlebox '15. New York, NY, USA: ACM, 2015, pp. 43–48. [Online]. Available: http://doi.acm.org/10.1145/2785989.2785997

[16] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proceedings IEEE INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation*, vol. 2, Mar. 1996, pp. 594–602 vol.2.