A Routing and Placement Scheme for Network Function Virtualization

Jorge Crichigno*,†, Diogo Oliveira‡, Mahsa Pourvali‡, Nasir Ghani‡, David Torres†

*Department of Integrated Information Technology, University of South Carolina, Columbia, SC - USA
†College of Engineering and Technology, Northern New Mexico College, Espanola, NM - USA
‡Electrical Engineering Department, University of South Florida, Tampa, FL - USA
jcrichigno@nnmc.edu, diogoo@mail.usf.edu, mpourvali@mail.usf.edu, nghani@usf.edu, davytorres@nnmc.edu

***Abstract*—Network Function Virtualization (NFV) and Software-Defined Networking (SDN) are two paradigms that have attracted much attention in the networking field. The first permits the implementation of Network Functions (NFs) on commodity servers located in datacenters. The second facilitates the management and routing of network flows by controllers. While recent work has mostly explored the use of NFV and SDN with the goal of minimizing the resources to satisfy a set of requested NFs, the application of these two paradigms has not been studied in scenarios where resources are limited. Those scenarios are typical when parts of the network or datacenters fail. Hence, this paper presents an optimization scheme based on integer linear programming (ILP) for the joint routing and placement of virtual NFs problem. Given a set of requests, each consisting of NFs and end points of the traffic flow, the objective of the scheme is the maximization of the number of NFs satisfied. At the same, the scheme minimizes both the routing and infrastructure costs to satisfy the requests. While numerical results demonstrate that the proposed ILP can be used in small/medium-sized networks, the paper also presents a low-complexity greedy heuristic approach for large networks.**

***Keywords*—Greedy algorithm; integer and linear programming; Network Function Virtualization (NFV).**

I. Introduction

Network Function Virtualization (NFV) is a paradigm by which network functions (NFs) such as firewall, access-control list, and intrusion prevention systems are executed on commodity servers at datacenters. By decoupling the physical network equipment from the functions that run on them, NFs can be dispatched to a service provider's datacenter as an instance of plain software. NFV thus does not only reduce operating and capital expense costs but also facilitates the deployment of new services and the selection of the most appropriate implementation of NFs among multiple NF providers [1].

Software Defined Networking (SDN) has attracted significant attention as the most relevant architecture for network management. SDN decouples the network control and forwarding functions. This permits the network control to become programmable and the underlaying infrastructure to become simple packet forwarding devices that can be programmed. With SDN, a controller or set of controllers maintain a global view of the network. Controllers are then responsible of the routing of flows from source to destination nodes [2].

The complementary relation between SDN and NFV is manifested in modern networks where a service provider receives requests composed of NFs subject to flow constraints. Each request is composed of a traffic flow, which must traverse the network from an ingress switch to an egress switch, and a set of NFs to be implemented at the service provider's datacenters (alternatively, there may be several competing service providers). The service provider must also route the requested traffic flow from the ingress switch to the egress switch, passing through those datacenters designated to satisfy the requested functions. Note that in this work, the order in which functions are applied to a traffic flow is not relevant. This applies to traffic such as large data flows traversing Science DMZs, where flows are typically subject to few NFs such as access-control lists and network flow monitoring which can be applied in an arbitrary order [3].

Previous work has focused on the minimization of resources needed for orchestration and placement of NFs [4], [5], [6], [7] under the condition that datacenters have enough resources to satisfy all requests. On the other hand, the maximization of the number of satisfied requests under the condition that datacenters may not have enough resources to satisfy all requests has not been investigated. Such *under-resourced* scenarios are typical when parts of the network fail, including links and datacenters, and the capacity to serve requests decreases.

This paper presents an integer linear program (ILP) for the joint routing and placement of NFs problem. The proposed scheme maximizes the number of requested NFs and at the same time minimizes the resources needed for both serving the requested NFs and routing each traffic flow. The ILP can generate optimal solutions for small/medium-sized networks in a reasonable amount of time and is amenable for deployment in NFV- and SDN-based networks. The paper also presents a low-complexity greedy heuristic approach.

The paper is organized as follows. Section II discusses related work. Section III formulates the problem and present the ILP and heuristic schemes. Section IV shows numerical results, and Section V concludes the work.

This work was partially supported by National Science Foundation of USA under Grant ACI 1541352.

II. Related Work

Coen et al. [5] presented an ILP to minimize the cost of placing virtual NFs into datacenters or nodes. The authors also presented multiple approximation algorithms. However, this model does not take routing into account. Addis et al. [4] proposed an ILP to minimize the number of CPUs used to satisfy NF requests. Due to the complexity of the scheme (large number of variables and constraints), the model was only applied to small instances. The authors also proposed a heuristic to achieve balance between traffic engineering and CPU minimization. Xia et al. [6] considered the placement of NFs in an environment with both optical and electronic elements. The proposed ILP minimizes the optical/electronic/optical conversions for network function virtualization chaining in packet/optical datacenters. The authors also presented a heuristic algorithm to solve the problem. Bouet et al. [7] considered a single service (deep packet inspection) for which a placement heuristic is presented. Gebert et al. [8] presented a scheme for the placement of mobile core network functions, where a service provider is required to provide optimal cellular coverage for large events.

To the best of the authors' knowledge, this paper is the first study that addresses the problem of maximizing the number of requested NFs while minimizing the routing and infrastructure costs to deploy NFs in scenarios where datacenters may not have enough resources to satisfy all requests. The proposed scheme is suitable for routing and placement of NFs in which the order of NFs applied to a traffic flow is flexible.

III. Joint Routing and Placement of NFs

A. Problem Formulation

The network is represented as a graph $G = (V, E)$, where V is the set of nodes and E the set of links. Each link $(i, j) \in E$ has an associated cost c^{ij} which quantifies the cost of using that link. The subset $D \subseteq V$ represents the set of datacenters where NFs are implemented. The set of NFs is denoted by F. A datacenter $d \in D$ implements a subset of functions $F_d \subseteq F$. Let R be the set of requests. Each request $r \in R$ is characterized by a 3-tuple (src_r, dst_r, F_r), which denotes the source and destination nodes of the flow and the set of functions $F_r \subseteq F$ the request r is interested in.

There is a number of resource types denoted by m; e.g., CPU, storage, memory. A datacenter $d \in D$ has a finite amount of resources $W_d = \{w_{d,1}, w_{d,2}, ..., w_{d,m}\}$. To implement a function $i \in F_d$, datacenter $d \in D$ employs $w^i_{d,1}, w^i_{d,2}, ..., w^i_{d,m}$ resources. This resource requirement is datacenter dependent, which mirrors the fact that some datacenters may specialize in the implementation of certain functions. The setup cost of locating an instance of a function $i \in F_d$ at datacenter d is c^i_d. An instance of function i in datacenter d can serve λ^i_d requests. To accommodate more requests, multiple instances of function i can be deployed at datacenter d; however, each instance consumes additional resources and requires additional setup cost.

The objective of the joint routing and placement of NFs problem under resource limitations is the maximization of the number of satisfied NFs requested by all requests while minimizing the routing and the NFs setup costs.

$$
\begin{aligned}
\max\ F \ &= \ w_1 \sum_{r \in R} \sum_{i \in F_r} \sum_{d \in D | i \in F_d} x^i_{r,d} - w_2 \sum_{d \in D} \sum_{i \in F_d} c^i_d y^i_d \\
&\quad - w_3 \sum_{r \in R} \sum_{(i,j) \in E} c^{ij} l^{ij}_r && (1) \\
\sum_{d \in D} x^i_{r,d} \ &\leq \ 1 \qquad r \in R, i \in F_r && (2) \\
x^i_{r,d} \ &\leq \ y^i_d \qquad r \in R, i \in F_r, d \in D | i \in F_d && (3) \\
\sum_{i \in F_d} w^i_{d,j} y^i_d \ &\leq \ w_{d,j} \qquad d \in D, r \in R, j \in \{1, 2, ..., m\} && (4) \\
\sum_{r \in R} x^i_{r,d} \ &\leq \ \lambda^i_d y^i_d \qquad d \in D, i \in F_d && (5) \\
\sum_{j:(i,j) \in E} l^{ij}_r \ - \ \sum_{j:(j,i) \in E} l^{ji}_r &= \begin{cases} -1; i = dst_r,\ src_r \neq dst_r \\ 1; i = src_r,\ src_r \neq dst_r \\ 0; \text{otherwise.} \quad i \in V, r \in R \end{cases} && (6) \\
\sum_{(d,j) \in E} l^{dj}_r \ &\geq \ x^i_{r,d} \qquad r \in R, i \in F_r, d \in D | i \in F_d && (7) \\
x^i_{r,d} \ &\in \ \{0, 1\} \quad r \in R, i \in F_r, d \in D | i \in F_d && (8) \\
y^i_d \ &\in \ \mathbb{Z}^+ \qquad d \in D, i \in F_d && (9) \\
l^{ij}_r \ &\geq \ 0 \qquad r \in R, (i, j) \in E && (10)
\end{aligned}
$$

Fig. 1. ILP for the joint routing and placement of NFs problem.

Fig. 1 shows the proposed ILP. Consider the variable definitions. The variable $x^i_{r,d}$, given by Constraint (8), indicates whether the function $i \in F_r$ requested by request $r \in R$ is implemented at datacenter $d \in D$. The variable y^i_d, given by Constraint (9), gives the number of instances of function $i \in F$ at node $d \in D$. The variable l^{ij}_r, given by Constraint (10), indicates whether the link $(i, j) \in E$ is used to route the traffic flow of request $r \in R$. While a variable l^{ij}_r is not constrained to $\{0, 1\}$, the ILP constraint matrix drives the value to zero or one. Eq. (1) is the objective function to be maximized, composed of three terms multiplied by their respective weight factors w_1, w_2, and w_3. The first term is the total number of requested NFs. The second term is the total cost to setup NFs at the multiple datacenters. The third term is the total routing cost. The second and third terms are negative, because maximizing a negative term is equivalent to minimizing it [9], [10]. Constraint (2) indicates that a function i requested by request r is serviced by at most one datacenter d. Since the objective of the ILP is the maximization of the sum of all variables $x^i_{r,d}$ (first term of Eq. (1)), the optimal solution will drive Constraint (2) to equality. Constraint (3) states that if request r is assigned to function i at datacenter d, then function i is located at d. Constraint (4) states that the aggregate amount of resources of type j used by all functions instantiated at datacenter d is limited by the total amount of resources $w_{d,j} \in W_d$, $j \in \{1, 2, ..., m\}$. Constraint (5) indicates that the total number of requests of function i served by datacenter d is at most the number of instances of i at d times the capacity λ^i_d of an instance of i. Constraint (6) is the flow conservation constraint. Constraint (7) guarantees that if a function i requested by request r is placed at datacenter d then the traffic flow will be routed through that datacenter.

B. ILP Complexity

Variables $x^i_{r,d}$ given by Constraint (8) are binary. Variables y^i_d given by Constraint (9) are integer. The objective function and constraints are linear. Hence, the model is an ILP and is NP-hard [11]. The upper-bounds of the number of Constraints (2), (3), (4), (5) are $|R||F|$, $|R||F||D|$, $|R||D||W_d|$, and $|F||D|$ respectively. The upper-bounds for Constraints (6) and (7) are $|V||R|$ and $|R||F||D|$. Thus, the upper-bound on the number of constraints is dominated by the product $|R||F||D|$. Similarly, the number of variables $x^i_{r,d}$, y^i_d, and l^{ij}_r given by Constraints (8), (9), and (10) are upper-bounded by $|R||F||D|$, $|F||D|$, and $|R||E|$. Therefore, the upper-bound of the number of variables is dominated by the product $|R||F||D|$ as well. Assuming that the model is applied to medium-sized networks (i.e., V is in the tens or few hundreds only), then the the number of variables and constraints is $O(|R||F||D|)$. If the product $|R||F||D|$ is not very large, the proposed ILP can be solved in a reasonable amount of time by branch and bound or other algorithms for ILP.

Algorithm 1 Greedy Routing and Placement of NFs

```
1. INPUT: $G(V,E), c^{ij} \forall (i,j) \in E, R, F, D$
2. OUTPUT: $x^i_{r,d}, y^i_d, l^{ij}_r$ values
3. set $x^i_{r,d} = 0$, $y^i_d = 0$, $l^{ij}_r = 0$ for all $r \in R$, $i \in F_r$, $d \in D$, $(i,j) \in E$
4. for all $r \in R$ do
5.     $D(r) = \{\}$
6.     $k = 1$
7.     for all $i \in F_r$ do
8.         $d_k$ = datacenter that implements $i$ at minimum cost and has enough resources
           to serve an additional request
9.         update resources of $d_k$
10.        update $y^i_{d_k}$
11.        set $x^i_{r,d_k} = 1$
12.        $D(r) = D(r) \cup d_k$
13.        $k = k + 1$
14.    end for
15. end for
16. for all $r \in R$ do
17.     $src = src_r$
18.     $C(r) = \{src\}$
19.     for $k = 1$ to $|D(r)|$ do
20.         $dst = d_k$
21.         if $d_k \ni C(r)$ then
22.             $SP = Dijkstra(src, dst)$
23.             set $l^{ij}_r = 1$ for all link $(i,j) \in SP$
24.             $C(r) = C(r) \cup d_k$
25.             $C(r) = C(r) \cup j$, for all datacenter $j \in SP$, $j \in D(r)$
26.         end if
27.         $src = dst$
28.     end for
29.     $dst = dst_r$
30.     $SP = Dijkstra(src, dst)$
31.     set $l^{ij}_r = 1$ for all $(i,j) \in SP$
32. end for
33. return $x^i_{r,d}, y^i_d, l^{ij}_r$
```

C. Greedy Heuristic

This section presents a greedy heuristic algorithm that can be applied to large networks, which is illustrated in Algorithm 1. Given the graph and input parameters, the algorithm returns the values for variables $x^i_{r,d}$, y^i_d and l^{ij}_r. The algorithm proceeds in two stages. The first stage (lines 4-15) computes the placement of NFs into datacenters. For each function i requested by $r \in R$, the algorithm selects the datacenter d_k that implements i at the lowest cost and has the resources to serve an additional request (line 8). Resources at datacenter d_k are updated (line 9) as follows: if there is an instance of function i at datacenter d_k with sufficient instance capacity, then the remaining capacity of that instance is decremented. If no instance of i is available to serve the request, then a new instance is created at a cost $c^i_{d_k}$. The resources at d_k are reduced by $w^i_{d_k,j}$, for all resource $w_{d,j} \in W_d$. The new instance can serve the current request and additional $\lambda^i_{d_k} - 1$ requests, and variable $y^i_{d_k}$ is also incremented (line 10). Variable x^i_{r,d_k} is set to one (line 11). Datacenter d_k is then added to $D(r)$, the set of datacenters serving request r.

For each $r \in R$, the second stage of the algorithm (lines 16-32) creates a path from the source node src_r, passing through all datacenters in $D(r)$, to the destination node dst_r. During the first iteration of the inner loop (lines 19-28), the algorithm connects the source node of the flow, src_r, to the first datacenter d_1 added to the set $D(r)$ during the previous stage. The algorithm used for the connection is Dijkstra's shortest path (line 22). For each link (i,j) along the shortest path, the variable l^{ij}_r is set to one (line 23). Also datacenter d_k is added to the set $C(r)$ (line 24), which includes the datacenters serving r that were already connected and will be in the path from node src_r to node dst_r. Any other datacenter $j \in D(r)$ that is in the shortest path between src and dst is also added to $C(r)$ (line 25). Line 25 avoids potential redundant connections which would have been added on next iterations of the inner loop. Lines 29-31 connect the destination dst_r with the last datacenter added to $C(r)$.

D. Greedy Heuristic Complexity

The proposed heuristic is a greedy algorithm. During the first stage, the greedy choice is executed by selecting the datacenter d that implements the function $i \in F_r$ at the lowest cost c^i_d (line 8). During the second stage, the choice to use the shortest path between two datacenters and between datacenters and end points represents a greedy approach (lines 22 and 30). By choosing the most obvious and immediate benefit only, the complexity of the algorithm is kept low. Since Dijkstra's algorithm is repeatedly executed in line 22 (double loop), stage 2 (lines 16-32) has higher complexity than stage 1 (line 4-15). Thus the running time is dominated by stage 2. In the worst case, Dijkstra's algorithm is called $|R||D(r)|$ times in line 22. Since $|D(r)| \leq |D|$, Dijkstra's algorithm is executed at most $|R||D|$ times. The Dijkstra's algorithm was implemented using binary heaps, therefore it has a running time of $O(|E|log|V|)$ [11]. The running time of the proposed heuristic, dominated by the second stage, is then $O(|R||D||E|log|V|)$.

IV. Numerical Results

For all simulations, $w_1 = 1000$ and $w_2 = w_3 = 1$. Here, the ILP solutions focus on the maximization of the number of satisfied NFs. The types of resources at a datacenter are three; i.e., for all $d \in D$, $W_d = \{w_{d,1}, w_{d,2}, w_{d,3}\}$. This set mirrors the most typical set of resources: CPU, memory, and storage. The amount of resources $w_{d,j}$ is randomly set to a number in the uniform range $\frac{100}{3} \leq w_{d,j} \leq 300$,

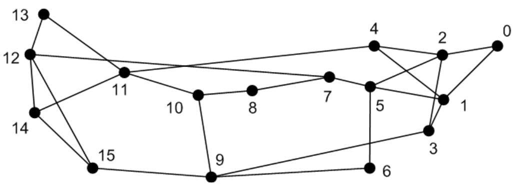

Fig. 2. NSF network topology.

for $j = 1, 2, 3$ and $d \in D$. The set F consists of five functions, $F = \{f_0, f_1, ..., f_4\}$. The amount of resources $w^i_{d,j}$ required to implement a function $i \in F_d$ is randomly set to a number in the uniform range $0 \leq w^i_{d,j} \leq 100$. These resource values produced *under-resourced* scenarios, where serving every requested NF may not be viable. The setup cost of locating an instance of a function $i \in F_d$ at datacenter $d \in D$ was uniformly distributed between $0 \leq c^i_d \leq 100$. Similarly, the instance capacity of a function i at datacenter d was uniformly distributed between $1 \leq c^i_d \leq 3$. The set of functions F_d implemented by a datacenter d was composed of three randomly selected functions from F.

The locations of datacenters are randomly selected among the nodes of the network topology shown in Fig. 2. For each generated request $r \in R$, the set of functions $F_r \subseteq F$ and the end points src_r and dst_r are also generated randomly.

Fig. 3 shows the results of simulated scenarios where each request was interested in all five functions of F; i.e., $F_r = F$ for all $r \in R$. The number of datacenters was 11. The values in the black curve were obtained by relaxing the ILP to a linear program (LP). This relaxation represents an upper-bound for the ILP and heuristic in the number of satisfied NFs. The red and blue curves are the values obtained with the ILP and greedy heuristic respectively. Note that the curves in Fig. 3(d), 3(e), and 3(f) reflect the gap between the values obtained with the ILP and greedy heuristic with respect to the bound obtained with the LP. These values are computed as follows:

$$\text{Gap} = \frac{ov_{LP} - ov_{alg}}{ov_{LP}} \tag{11}$$

where ov_{LP} is the optimal LP value, and ov_{alg} is the optimal value obtained with ILP or greedy heuristic.

According to Fig. 3(a) the number of NF requests are only fully satisfied when the number of requests is one (only five NF are requested, as the only request r is interested in five functions). As the number of requests increases, the number of satisfied NFs also increases, but at slower pace. For example, given $|R| = 5$ (25 NFs requested), the number of satisfied NFs is less than 20. Even under an under-resourced environment, a larger number of requests increases the opportunity to satisfy a larger number of NFs. While the number of satisfied NFs obtained by the ILP is the same as that by the LP (with the exception when $|R| = 13$, see Fig. 3(a)), the gap between the the greedy and the LP increases to more than 40% when $|R| = 15$ as shown in Fig. 3(d). Figs. 3(b) and 3(e) indicate that the ILP solutions cost approximately up to 40% more than those with the LP scheme, while those with the greedy scheme cost up to 250% more. Figs. 3(c) and 3(f) show that the gap in the routing cost between the LP and the ILP is less than 20%, while that of the greedy is up to approximately 120%. Note in Figs. 3(e) and 3(f) how the gap between the ILP and the LP remains approximately flat as the number of requests increases. However, it should be highlighted that the optimal values by the LP may not constitute feasible solutions, as the decision variables of the LP can take fractional values.

Figs. 4(g)-(l) illustrate numerical results for scenarios with 13 requests where each request is interested in one function. Fig. 4(g) shows that the ILP is able to satisfy all requests once the number of datacenters reaches five. While adding more datacenters do not have any effect in the number of satisfied NFs, they do reduce the NF deployment and routing costs by the ILP and the LP as seen in Figs. 4(h) and 4(i). The reason is that function instances can be located at a lower costs in the additional datacenters. Figs. 4(j)-(l) show the gap between the ILP and the LP and between the greedy heuristic and the LP for the results of Figs. 4(g)-(i). The gap between the ILP and the LP is zero when the number of datacenters is three or more. Fig. 4(l) indicates that the routing cost of the ILP solutions were lower than those by the LP bound. However, this result must be considered jointly with that of Fig. 4(k), where the NF deployment cost of the solutions by LP are lower than those by ILP. It is important to remark the fractional values for variables $y^i_d, d \in D, i \in F_d$ obtained by LP.

Figs. 4(a)-(f) show numerical results for scenarios with a small number of datacenters (three) and 15 requests. On the other hand, Figs. 4(g)-(l) show numerical results for scenarios with a large number of datacenters (11) and 15 requests. Some conclusions can be drawn from these different scenarios. As seen in Fig. 4(a), the number of satisfied requests increases as the number of functions per request increases for solutions using the LP and the ILP. The NF deployment cost in Fig. 4(b) also increases in the solutions by the LP and the ILP. However, notice that the routing cost produced by them is mostly flat in Fig. 4(c). This indicates that routing is less sensitive to scenarios with a small number of datacenters, as function instances are centralized in a small number of datacenters. In the contrasting scenario with 11 datacenters, the number of satisfied NFs also increases with the number of functions per request. Note in Fig. 4(g) and 4(j) that the gap between the ILP and the LP is zero. The corresponding NF deployment cost in Fig. 4(h), while increasing slightly with the number of functions per request (i.e., the increase in NF deployment cost is moderate when changing the number of functions per request from three to five), shows a smaller increase rate than that of Fig. 4(b). The reason here is that the large number of datacenters implementing instances of functions can serve additional requested NFs without a need to substantially increase the number of instances. Fig. 4(i) shows that the routing cost increases substantially in solutions using the LP and the ILP when the number of functions per request increases. Thus, when the number of datacenters is large, routing becomes more sensitive because the set of functions a request r may be interested in can be implemented in several datacenters across the network. The associated traffic flow must then traverse the network from the source node src_r

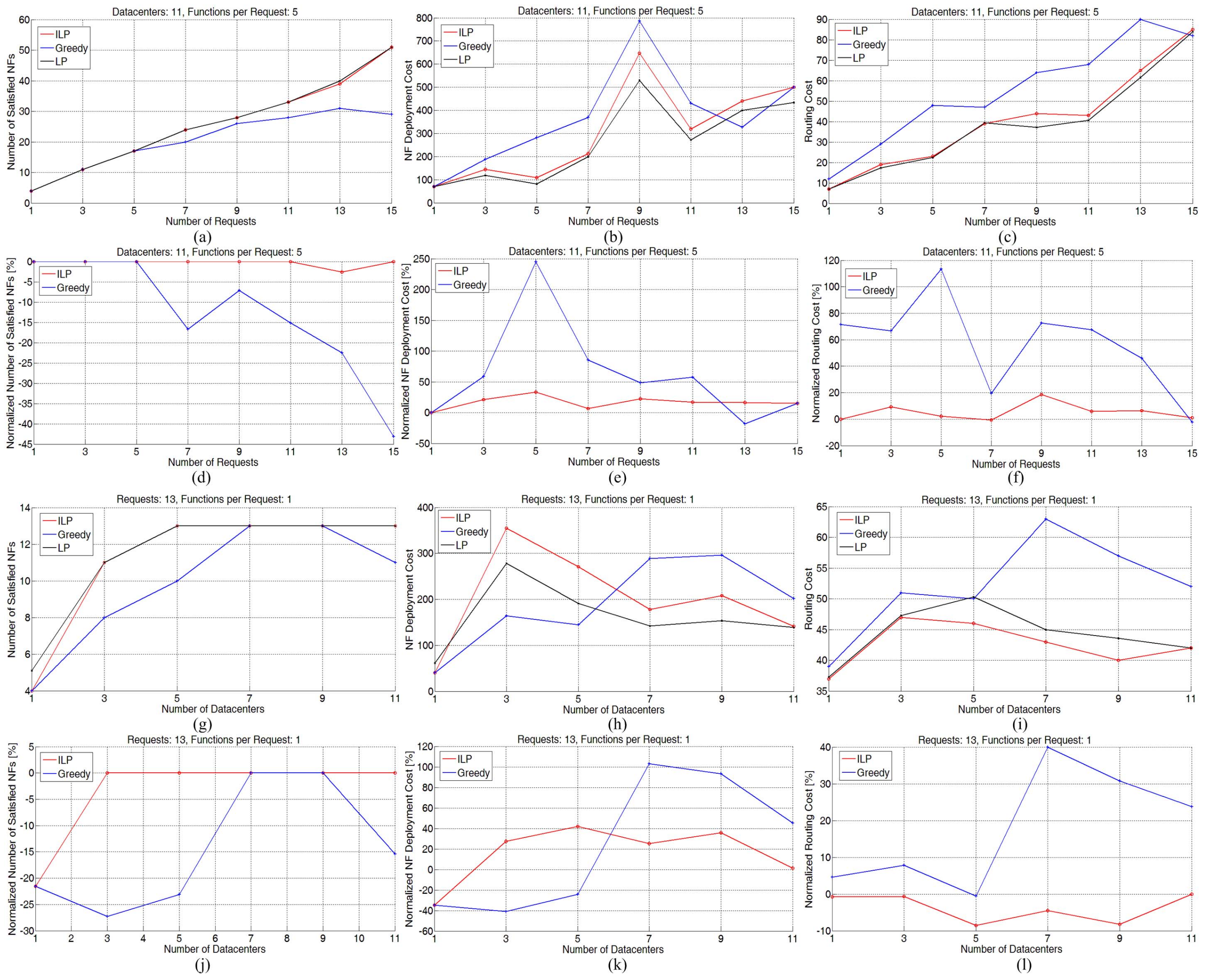

Fig. 3. Numerical results. The set of functions F consists of five functions. For (a)-(f), the number of datacenters and functions per request are 11 and five respectively. (d) illustrates the gap between the ILP and the LP curves (red) of (a), and between the greedy and LP curves (blue) of (a), computed according to Eq. (11). Similarly, (e) and (f) are the gaps observed in (b) and (c). For (g)-(l), the number of requests and functions per requests are 13 and one. (j)-(l) represent the gap corresponding to the curves in (g)-(i).

passing through all datacenters serving r to the destination node dst_r. Fig. 4(j) shows that the gap between the ILP and the LP solutions for the the number of satisfied NFs is zero. The corresponding NF deployment and routing costs demonstrate a gap of less than 15% and 10%. The greedy heuristic has a gap of up to 45%, 30% and 33% in the number of satisfied NFs, NF deployment cost and routing cost respectively.

V. Conclusion

This paper presents an optimization scheme for the joint routing and placement of virtual network functions (NFs) problem. The proposed integer linear program (ILP) maximizes of the number of satisfied NFs while at the same time minimizes the resources needed for both serving the requested NFs and routing each traffic flow. To the best of the authors' knowledge, maximizing the number of satisfied requests for the joint problem, under the condition that datacenters may not have enough resources to satisfy all requests, has not been previously studied. This *under-resourced* scenario is typical when network components or datacenters fail. Additionally, the proposed ILP considers heterogeneous environments where (a) the set of functions implemented by datacenters needs not to be the same at each datacenter, and (b) the cost and resources required to implement a virtual network function are datacenter-dependent. Numerical results show that the ILP generates optimal solutions for small/medium-sized networks in a reasonable amount of time. The results also show a small gap in the number of satisfied NFs (few percentage points) between the ILP and the corresponding LP relaxation. Additionally, the paper also presents a low-complexity greedy heuristic approach amenable for large networks.

The authors of the paper are investigating the application of the proposed scheme to several scenarios with different

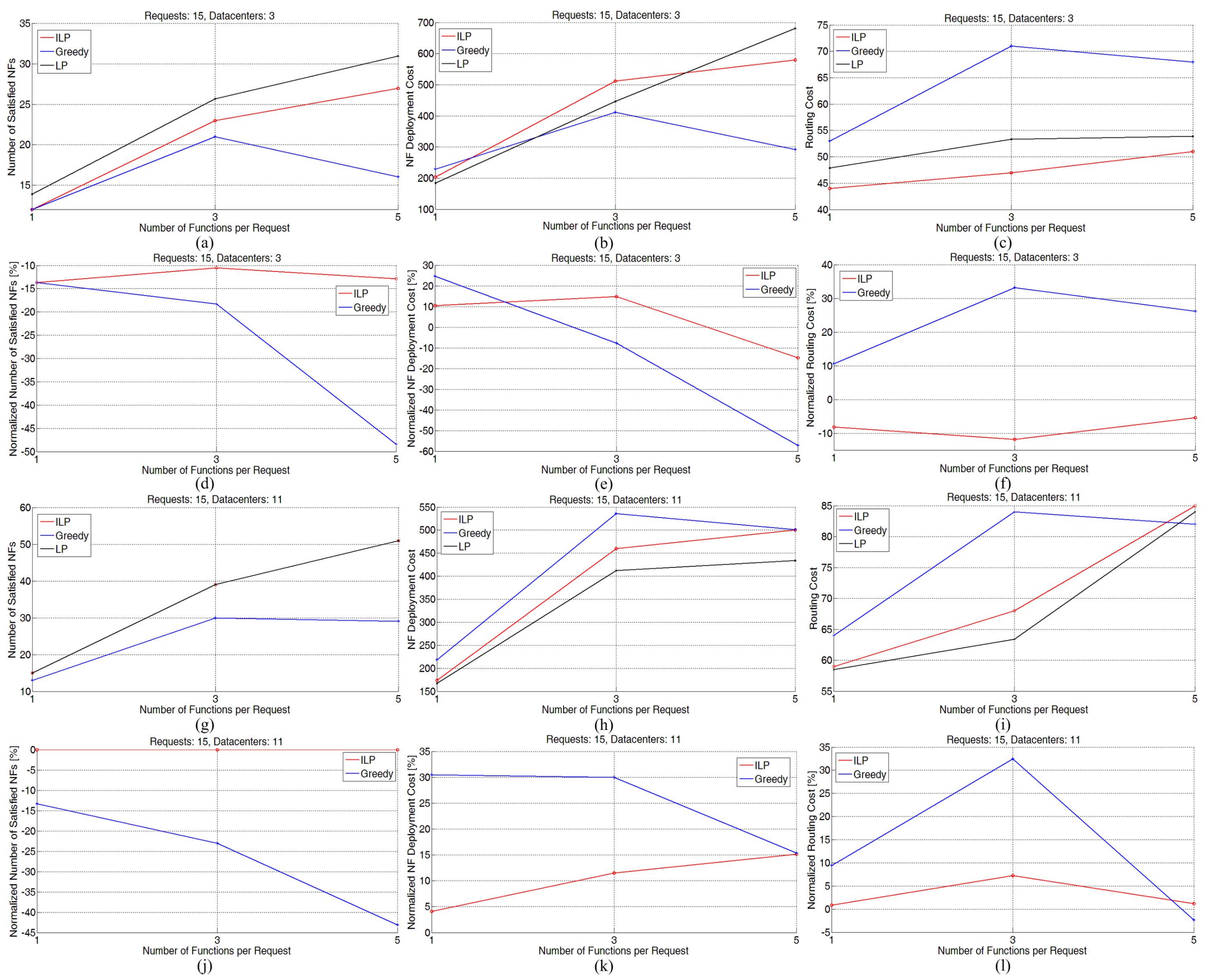

Fig. 4. Numerical results. The set of functions F consists of five functions. For (a)-(f), the number of requests and datacenters are 15 and three respectively. (d) illustrates the gap between the ILP and the LP curves (red) of (a), and between the greedy and LP curves (blue) of (a), computed according to Eq. (11). Similarly, (e) and (f) are the gaps observed in (b) and (c). For (g)-(l), the number of requests and datacenters are 15 and 11. (j)-(l) represent the gap corresponding to the curves in (g)-(i).

parameter values, including number of requests, datacenters, resource requirements, and number of functions. The scalability of the ILP with respect to the above parameters is also being studied. Finally, the implementation of the proposed scheme using ONOS SDN [12] is being evaluated.

References

[1] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," *IEEE Communications Survey and Tutorial*, Vol. 18 (1), 2016.

[2] D. Kreutz, F. Ramos, P. Verssimo, C. Rothenberg, S. Azodolmolky, S. Uhlig, "Software-Defined Networking: A Comprehensive Survey,", *Proceedings of the IEEE*, Vol. 103 (1), January 2015.

[3] Energy Science Network, [Online]. Available: http://fasterdata.es.net/science-dmz

[4] B. Addis, D. Belabed, M. Bouet, S. Secci, "Virtual Network Functions Placement and Routing Optimization," *IEEE International Conference on Cloud Networking*,Niagara Falls, Canada, October 2015.

[5] R. Cohen, L. Lewin-Eytan, J. Naor, D. Raz, "Near Optimal Placement of Virtual Network Functions," *IEEE International Conference on Computer Communications*, Hong Kong, April 2015.

[6] M. Xia, M. Shirazipour, Y. Zhang, H. Green, A. Takacs, "Network Function Placement for NFV Chaining in Packet/Optical Datacenters," *Journal of Lightwave Technology*, Vol. 33, Issue 8, April 2015.

[7] M. Bouet, J. Leguay, V. Conan, "Cost-Based Placement of Virtualized Deep Packet Inspection Functions in SDN," *Military Communications Conference*, San Diego, CA, November 2013.

[8] S. Gebert, D. Hock, T. Zinner, M. Hoffmann, M. Jarschel, E.D. Schmidt, R.P. Braun, C. Banse, and A. Kopsel, "Demonstrating the Optimal Placement of Virtualized Cellular Network Functions in Case of Large Crowd Events," *ACM SIGCOMM Conference*, Chicago, IL, August 2014.

[9] J. Crichigno, N. Ghani, J. Khoury, W. Shu, M. Wu, "Dynamic routing optimization in WDM networks," *IEEE Globecom Conference*, Miami, FL, December 2010.

[10] J. Crichigno, W. Shu, M. Wu, "Throughput optimization and traffic engineering in wdm networks considering multiple metrics," *IEEE ICC Conference*, Cape Town, South Africa, May 2010.

[11] T. Cormen, C. Leiserson, R. Rivest, C. Stein, "Introduction to Algorithms," 2nd Edition, McGraw Hill, 2001.

[12] Open Network Operating System (ONOS) SDN, [Online]. Available: http://onosproject.org/