On Sensitive and Weighted Routing and Placement Schemes for Network Function Virtualization

Diogo Oliveira, Jorge Crichigno and Nasir Ghani

***Abstract*—Virtualization is a fast-growing technology that is being widely adopted to help improve network and datacenter resource manageability and usage optimization. However, given increasing deployments, new challenges are starting to arise, e.g., such as management complexity. Hence in order to deliver a higher degree of service provisioning flexibility, two key technologies have attracted attention, namely network function virtualization (NFV) and software defined networking (SDN). The former enables the implementation of network functions (NFs) via top-of-the-shelf commodity servers in datacenters. The latter decouples the data and control planes, centralizing flow rules definitions in a controller system to facilitate management and routing. Although recent NFV studies have focused on minimizing resource usage to satisfy a set of requested NFs, they do not consider scenarios with limited resources. Hence this paper presents an optimization-based solution for the joint routing and placement of virtual NFs. In particular, the scheme tries to maximize the number of satisfied requests as well as well minimize routing and deployment costs. The model also introduces weighting factors to allow operators to select cost preferences. However findings indicate that the proposed optimization solution can only be solved for smaller networks. Hence a more scalable greedy heuristic scheme is also developed.**

***Index Terms*—Greedy algorithm; integer linear programming (ILP); network function virtualization (NFV).**

I. INTRODUCTION

NETWORK function virtualization (NFV) is a paradigm that allows network functions (NFs) to be executed on commercial-of-the-shelf (COTS) commodity servers, e.g., such as firewalls, load balancers, address translation boxes, etc. Namely, it decouples physical network devices from the functions that run on them, allowing such functions to be executed as software instances within datacenters. As such, NFV reduces capital and operational expenses by replacing embedded devices and facilitating the deployment and management of networking services [1].

Meanwhile, software defined networking (SDN) is being widely adopted due to its dynamic traffic flow management capabilities. This methodology decouples the data and control planes, relegating decision-making capacity to a controlling system called the SDN controller. As such, network control becomes much more flexible, and the underlying infrastructure can be reduced to a layer of programmable packet forwarding devices. Namely, SDN controllers can build global network views and directly compute traffic flow rules for routing packets from source to destination nodes [2].

In general, NFV and SDN concepts have an orthogonal relationship. However, even though these two paradigms do not depend upon each other, they are still very complementary. Namely, the former dynamically deploys NFs as software instances, whereas the latter steers traffic to datacenters hosting the desired NFs associated with a request. Now each service request is comprised of a source and destination pair, a set of requested NFs and a load rate. Hence a service provider instantiates the pertinent NFs (associated with the network services that are offered) in the datacenters and defines the proper flow rules in the SDN controller(s). This procedure helps establish the traffic flow between the source and the destination. Now clearly each traffic flow must pass through the datacenters hosting the requested NFs at the minimum data rate requested. However, the *sequence* in which desired functions are applied to a traffic flow is not considered here, i.e., service function chaining (SFC).

To date, existing studies have focused on the minimization of resources required for orchestration and placement of NFs under the assumption that datacenters have infinite resources to satisfy all requests [3], [4], [5], [6]. However, in resource constrained environments, the underlying physical infrastrurcture may not be able to satisfy all requests, i.e., during periods of heavy load or after large failure events. To the best of the authors' knowledge, the only known work on NF placement (to maximize the number of satisfied requests) in resource-constrained environments is presented in [7]. However this initial study does not consider link capacity constraints, link load balancing, or weighting factor selection/adjustment.

In light of the above, this paper presents a novel integer linear programming (ILP) solution for weighted joint routing and placement of NFs, i.e., termed as the *Minimized Link Load* ILP (MLL-ILP) scheme. The proposed scheme tries to maximize the number of requested NFs, and also minimize the deployment cost, routing cost and link load. Additionally, the scheme implements weighting factors for all three minimization costs to allow operators to fine tune NF placement and routing according to their operational needs. However, the MLL-ILP formulation is still intractable for larger network sizes. As a result, a greedy heuristic scheme is also developed to improve scalability, i.e., termed as the *Minimized Link Load Greedy* (MLL-GR) scheme.

This paper is organized as follows. First, Section II discusses some related work on NF placement, and then Section III

A preliminary version of this paper has been presented at the 2017 40th International Conference on Telecommunications and Signal Processing (TSP) [7].

D. Oliveira and N. Ghani are with the Department of Electrical Engineering, University of South Florida, Tampa, FL, 33620.
E-mail: diogoo@mail.usf.edu, nghani@usf.edu

J. Crichigno is with the College of Engineering and Computing, University of South Carolina, Columbia, SC, 29208.
E-mail: jcrichigno@sc.edu

formulates the proposed ILP and heuristic algorithms. Next, Section IV presents some detailed performance evaluation results, and Section V highlights the overall conclusions.

II. Related Work

As noted earlier, NFV is a very recent technology, the first white paper on which was published by the European Telecommunications Standards Institute (ETSI) in October 2012. However since then many researchers have studied the NF placement problem. For example, Coen et al. [4] present an ILP to minimize NF deployment cost, making use of multiple approximation techniques. However, this work does not consider routing costs. Following the minimization model, Addis et al. [3] define an ILP scheme to reduce the number of processors used to satisfy NF requests. However, this solution can only be applied to a small number of instances due to its complexity, i.e., high numbers of variables/constraints. Focusing on heterogeneous Layer 1 networks, Xia et al. [5] propose a method to minimize NF placement cost in hybrid networks comprised of optical and electronic network elements, i.e., optical-to-electronic and electronic-to-optical conversion costs. As a complementary solution, the authors also present a heuristic algorithm. Finally, Gebert et al. [6] present a heuristic NF placement scheme to allow service providers to optimize cellular coverage for large events.

However, to the best of the authors' knowledge, the only work which addresses the problem of maximizing the number of requested NFs while minimizing infrastructure (node, link) load and routing costs is presented in [7]. This technique is particularly germane in scenarios where datacenters may not have enough resources to satisfy all requests. Extending upon this, the work herein proposes additional schemes that incorporate link capacity constraints as well as load balancing. Furthermore, the proposed solution can also be adapted to suit provider needs. These flexible characteristics allows providers to fine tune NF placement according to their requirements, i.e., both in terms of routing cost and NF deployment cost. Additionally, link capacity constraints and load balancing features deliver a more realistic solution, i.e., since both physical node and transmission link resources can be limited in real-world settings. These solutions are now presented.

III. Weighted Joint Routing and Placement of NFs

As mentioned above, most NF placement schemes focus on minimizing placement cost by assuming unlimited network resources to satisfy all requests. Conversely, this paper looks at resource-limited scenarios and proposes two schemes (MLL-ILP and MLL-GR) to maximize the number of satisfied requests and reduce infrastructure and routing costs. Additionally, each cost has appropriately-defined weighting factors to create an adaptive model. For example if a provider has limited physical resources but sufficient link and routing capacity, then deployment cost can be given higher preference. On the other hand, if link capacities are lower and/or hops and delay times are higher, then reduced routing costs may be more favorable. Hence the proposed schemes allow providers to achieve a proper tradeoff between deployment and routing costs. Consider the details now.

A. Optimization Model

The network topology is represented as a graph $G = (V, E)$, where V is the set of nodes and E the set of links. Each link $(i, j){\in}E$ also has an associated cost c^{ij} and capacity b^{ij}, which quantifies the cost of using that link and its link capacity, respectively. Meanwhile, a subset of nodes $D \subseteq V$ represents the set of datacenters where NFs are implemented, and the set of NFs is denoted by F. A given datacenter $d{\in}D$ implements a subset of functions $F_d{\subseteq}F$. Furthermore, R represents the set of requests. Namely, each request $r{\in}R$ is characterized by a 4-tuple (src_r, dst_r, F_r, b_r), which denotes the source and destination nodes of the flow, the set of requested functions $F_r{\subseteq}F$, and the minimum available link capacity, respectively.

The customizable number of resource types is also denoted by an integer m. For example, m=3 can refer to processor, storage and memory. It is also assumed that a datacenter $d{\in}D$ has a finite amount of resources W_d=$\{w_{d,1}, w_{d,2}, ..., w_{d,m}\}$. Hence 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 reflects the fact that some datacenters may specialize in supporting certain functions. Also, the setup cost of locating an instance of a function $i{\in}F_d$ at datacenter d is c^i_d, and an instance of function i at datacenter d can serve λ^i_d requests. In order to accommodate more requests, multiple instances of function i can also be deployed at datacenter d. However each instance will consume additional resources and entails added setup cost.

Overall, the MLL-ILP objective function tries to maximize the total number of satisfied NFs and is composed of four terms weighted by their factors, i.e., w_1, w_2, w_3 and w_4:

$$\begin{aligned}\text{max F} \quad = \quad & w_1 \sum_{r\in R} \sum_{i\in F_r} \sum_{d\in D|i\in F_d} x^i_{r,d} - w_2 \sum_{d\in D} \sum_{i\in F_d} c^i_d y^i_d \\ & -w_3 \sum_{r\in R} \sum_{(i,j)\in E} c^{ij} l^{ij}_r - \alpha w_4\end{aligned} \tag{1}$$

The first term represents the total number of requested NFs. Meanwhile the second term is the total cost to setup/deploy NFs at the various datacenters. The third term is the total routing cost, and the fourth term represents the maximum overall link load. Note that the second, third and fourth terms are negative, since maximizing a negative term is equivalent to minimizing it [8], [9]. Also, the setup cost is directly related to the number of satisfied functions and number of instances of each function. Hence the second term (representing setup or deployment cost) depends upon the number of NFs per request and on how many datacenters are used to host these NFs. On the other hand, the third and fourth terms (representing routing and maximum link load costs, respectively) depend upon the number of requests and number of links used. These latter two terms are independent of the number of NFs and NF instances.

Next, Eqs. 2 to 12 list the complete set of model constraints:

$$x^i_{r,d} \quad \in \quad \{0, 1\} \quad r \in R, i \in F_r, d \in D|i \in F_d \tag{2}$$

$$y^i_d \quad \in \quad Z^+ \quad d \in D, i \in F_d \tag{3}$$

$$l_r^{i,j} \in \{0,1\} \quad r \in R, (i,j) \in E \tag{4}$$

$$0 \geq \alpha \leq 1 \tag{5}$$

$$\sum_{d \in D} x_{r,d}^i \leq 1 \quad r \in R, i \in F_r \tag{6}$$

$$x_{r,d}^i \leq y_d^i \quad r \in R, i \in F_r, d \in D | i \in F_d \tag{7}$$

$$\sum_{i \in F_d} w_{d,j}^i y_d^i \leq w_{d,j} \quad d \in D, r \in R, j \in \{1,2,...,m\} \tag{8}$$

$$\sum_{r \in R} x_{r,d}^i \leq \lambda_d^i y_d^i \quad d \in D, i \in F_d \tag{9}$$

$$\sum_{j:(i,j)\in E} l_r^{ij} - \sum_{j:(j,i)\in E} l_r^{ji} = \begin{cases} -1; i = dst_r, src_r \neq dst_r \\ 1; i = src_r, src_r \neq dst_r \\ 0; \text{ otherwise. } i \in V, r \in R \end{cases} \tag{10}$$

$$\sum_{(d,j)\in E} l_r^{dj} \geq x_{r,d}^i \quad r \in R, i \in F_r, d \in D | i \in F_d \tag{11}$$

$$\sum_{r \in R} l_r^{i,j} b_r \leq \alpha b_{i,j} \quad \{i,j\} \in E \tag{12}$$

The above constraints are now detailed further. Foremost, variable $x_{r,d}^i$ in Constraint 2 indicates whether function $i{\in}F_r$ requested by request $r{\in}R$ is implemented at datacenter $d{\in}D$. Meanwhile, variable y_d^i in Constraint 3 represents the number of instances of function $i{\in}F$ at node $d{\in}D$. Variable $l_r^{i,j}$ in Constraint 4 indicates whether link $(i,j){\in}E$ is used to route traffic flow for request $r{\in}R$, i.e., it is binary. Also, Constraint 5 limits α between 0 and 1 since this variable represents the highest link usage ratio, i.e., sum of all b_r using a link $(i,j){\in}E$ divided by link capacity $b_{i,j}$. Constraint 6 indicates that a function i requested by request r is serviced by at most one datacenter d. Since the objective of the ILP is to maximize the sum of all variables $x_{r,d}^i$ (first term in Eq. 1), the optimal solution will drive Constraint 6 to equality. Meanwhile Constraint 7 states that if request r is assigned to function i at datacenter d, then function i is located at d. Constraint 8 also states that the aggregate amount of type j resources used by all functions instantiated at datacenter d is limited by the total amount of resources $w_{d,j}{\in}W_d, j{\in}\{1,2,...,m\}$. Similarly, Constraint 9 indicates that the total number of requests for function i served at datacenter d is at most the number of instances of i at d times the capacity λ_d^i of an instance i. Meanwhile, Constraint 10 ensures necessary flow conservation. Finally, Constraint 11 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.

Now one of the key goals of MLL-ILP is also to avoid link overload or at least minimize link load. Therefore the fourth term in Eq. 1 introduces a link load minimization function variable α, which is linked to Constraint 12. Namely, this constraint checks if the sum of all links (i,j) used by a traffic flow associated with request r times the load b_r demanded by request r is bounded by the product of the link capacity $b_{i,j}$ times α. As a result, traffic flows are associated with links that have little/no load profile (higher available capacity).

Note that a non-link load balancing optimization scheme can also be defined by eliminating α, i.e., α=0. This instance is the same as the scheme presented in [7] and is termed as the *standard routing and placement ILP* (STD-ILP) scheme.

B. MLL-ILP Complexity

Since the objective function and all constraints are linear, and all variables are either integer or binary, the MLL-ILP problem is NP-hard [10]. Furthermore, the complexity of MLL-ILP can be determined by the number of variables that the scheme utilizes. Namely, the number of variables $x_{r,d}^i$, y_d^i and $l_r^{i,j}$, given by Constraints 2, 3 and 4 is upper-bounded by $|R||F||D|$, $|F||D|$ and $|R||E|$, respectively. Also, Constraint 5 refers to a single variable, α. Hence the upper-bounds for the number of variables in Constraints 6, 7, 8 and 9 are $|R||F|$, $|R||F||D|$, $|R||D||W_d|$ and $|F||D|$, respectively. Meanwhile, the upper-bounds on variable counts for Constraints 10 and 11 are $|V||R|$ and $|R||F||D|$. Also, the upper-bound for Constraint 12 is $|V||R|{+}1$. Hence the upper-bound for the total number of variables is dominated by the product $|R||F||D|$. Now in general it is very difficult to pre-define limits on the number of requests or functions to ensure ILP convergence. However for small to medium network topologies, the proposed ILP can still be solved in a reasonable amount of time, i.e., low hundreds of nodes.

C. MLL-GR Heuristic

Although the MLL-ILP approach can provide an optimal placement solution, its scalability is limited for large/complex scenarios with many variables. It is here that heuristic schemes can be developed to provide non-optimal solutions. Now a broad range of strategies are available here, i.e., such as genetic algorithms, particle swarm optimization, simulated annealing, greedy heuristics, etc. Accordingly, the latter approach is chosen here to solve the NF placement problem for larger network sizes, as shown in Fig. 1.

Overall, the proposed heuristic scheme implements a two-stage solution. Namely, given a graph and input parameters (akin to MLL-ILP; source, destination, set of NFs and minimum link capacity), the algorithm returns the values for variables x^i, r, d, y_d^i, $l_r^{i,j}$ and α. Foremost, the first stage of the algorithm in Fig. 1 (lines 4-15) places the NFs at datacenters to reduce deployment cost. For each function i requested by $r{\in}R$, the scheme selects the datacenter d_k that implements i with the lowest setup cost (and at the same time has sufficient resources to host upcoming requests, line 8). Once a NF is placed, the resources at datacenter d_k are updated accordingly as follows: if there is an instance of NF i at datacenter d_k with enough instance capacity to satisfy request r, the remaining capacity is decremented by 1. Otherwise, if there is no instance of i to serve request r, another instance is created at cost $c^i_{d_k}$. Either way, the available resources $w_{d_k,j}$ (j=$1,2,...,m$, where m is the number of resources) at d_k are reduced by $w^i_{d_k,j}$ (line 9). Each newly-created instance can serve $\lambda^i_{d_k}$-1 requests. Carefully note that serving a request does not incur any cost, i.e., only new instantiations of i do. Additionally, for each new instance of i, the total number of instances $y^i_{d_k}$ at d_k are incremented by 1 (line 10) and for each deployed NF, x^i_{r,d_k} is set to 1 (line 11). To conclude NF placement (first stage), datacenter d_k is also added to the subset of datacenters that serve request r, $D(r)$.

Meanwhile, the second stage in Fig. 1 (lines 16-32) computes the shortest path between the source and destination nodes that passes through all datacenters $d_k \in D(r)$. Initially, the algorithm connects the source node src_r to the first datacenter d_1 of $D(r)$. In particular the constrained Dijkstra's shortest path algorithm is used here to compute the connection route (line 22). This approach first verifies whether each link has enough capacity to support the requested b_r. All variables $l^{i,j}_r$ along the path are then set to 1 (line 23), and datacenter d_1 is also added to the subset $C(r)$ (line 24). This subset defines the datacenters that serve request r and are already connected to their respective neighbors within the src_r and dst_r path. Now if there is another datacenter $j \in D$ in the path between src and dst, it is also added to $C(r)$ (line 25). This method avoids duplicate path computation for datacenters yet to be analyzed in upcoming iterations. Before returning to the beginning of the loop, the src variable is replaced by the destination datacenter dst (line 27), which was initially set to d_k (line 20). At the top of the loop, dst is reset to the next datacenter $d_k \in D(r)$. Hence a shortest path is computed between the previous destination and the following datacenter, i.e., in the second iteration a path is computed between d_1 and d_2. In the two final steps, a shortest path is computed between the last datacenter and the destination dst (line 30), and all links $l^{i,j}_r$ within that traffic flow are set to 1 (line 31).

Akin to the STD-ILP scheme detailed at the end of Section III-A, a non-load balancing greedy heuristic scheme is also defined, i.e., α=0. Again this solution is similar to the approach presented in [7] and is termed as the *standard routing and placement greedy* (STD-GR) heuristic.

D. MLL-GR Complexity

The run-time complexity of the heuristic scheme is now analyzed. Conceptually, the greedy algorithm tries to lower complexity by finding the first acceptable solution. Therefore in the first stage it selects the datacenter d that can implement function $i \in F_r$ at the lowest cost c^i_d (line 8). Note that NF placement here is done in a serialized manner, i.e., each NF placement is independent of the following NFs requirements. Meanwhile, the second stage of the heuristic has higher complexity, i.e., since link-weighted shortest paths are computed between the datacenters and endpoints. To achieve this, the routing cost is defined as the sum of the setup cost of all links associated with a traffic flow and each respective link capacity/request load ratio, as shown in Eq. 13.

```
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 $r \in R, i \in F_r, d \in D, (i,j) \in E$
   {BEGIN FIRST STAGE}
4: for all $r \in R$ do
5:    $D(r) = \{\}$
6:    $k = 1$
7:    for all $i \in F_r$ do
8:       $dk$ = 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
   {END FIRST STAGE}
   {BEGIN SECOND STAGE}
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 $d_k \notin C(r)$ then
22:         $SP = constrained\_Dijkstra(src, dst)$
23:         set $l^{ij}_r = 1$ for all link $(i,j) \in SP$
24:         $C(r) = C(r) \cup d_k$
25:         $C(r) \cup j$, for all datacenter $j \in SP, j \in D(r)$
26:      end if
27:      $src = dst$
28:   end for
29:   $dst = dst_r$
30:   $SP = constrained\_Dijkstra(src, dst)$
31:   set $l^{ij}_r = 1$ for all $(i,j) \in SP$
32: end for
   {END SECOND STAGE}
33: return $x^i_{r,d}, y^i_d, l^{ij}_r$
```

Fig. 1: MLL-GR Algorithm

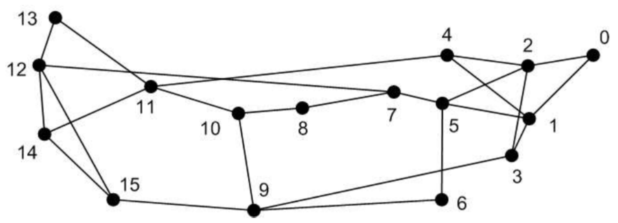
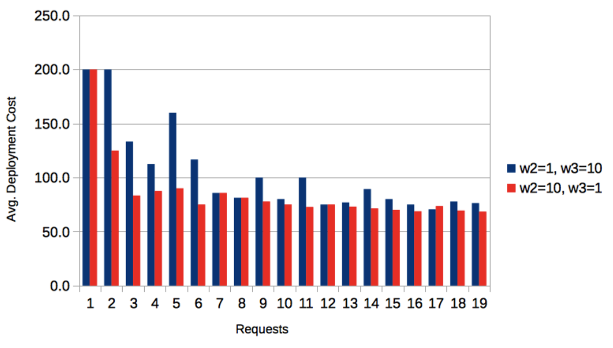


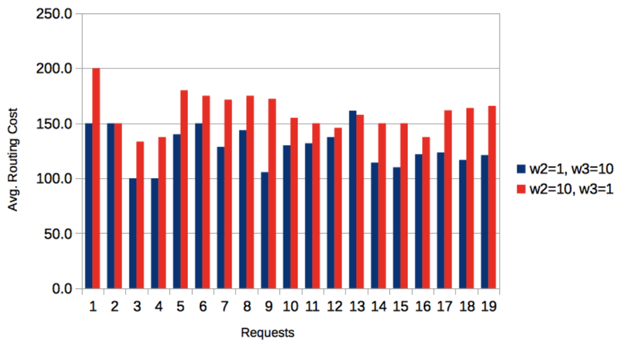
Fig. 2: NSF network topology

$$\sum_{(i,j)\in E} c^{ij} + b_r/b_{ij} \tag{13}$$

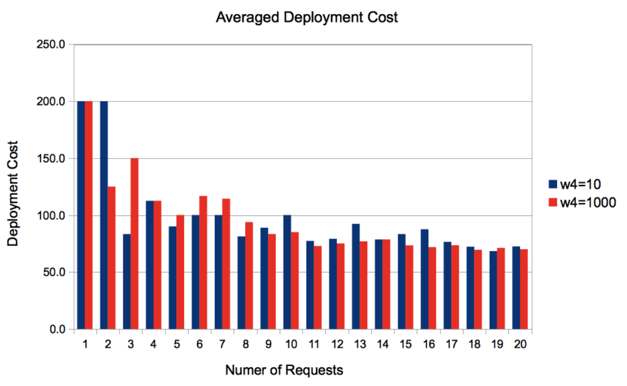
Since the constrained Dijkstra algorithm is executed within a double loop in Fig. 1, the second stage poses higher complexity than the first stage. Hence the MLL-GR run-time complexity is dominated by the second stage. In particular, this stage is invoked $|R||D|$ times, $D(r) \leq D$. Assuming a binary heap Dijkstra implementation complexity of $O(|E|log|V|)$ [10], the overall run time is of order $O(|R||D||E|log|V|)$.



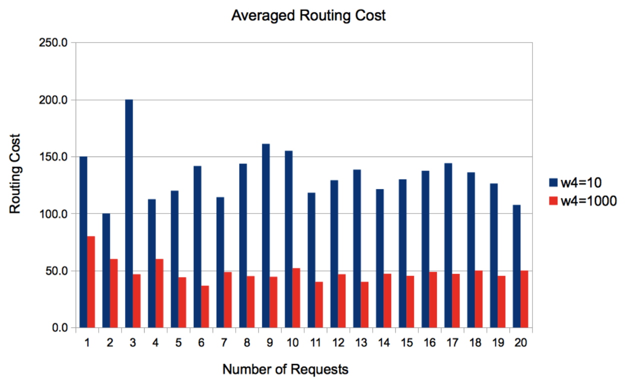
(a) Average deployment cost for differing w_1 and w_2



(b) Average routing cost for differing w_1 and w_2



(c) Average deployment cost for varying w_4



(d) Average routing cost for varying w_4

Fig. 3: First testcase (weight selection)

IV. PERFORMANCE EVALUATION

In order to evaluate the proposed NF placement schemes, a realistic network topology and a set of metrics must be defined. Hence the NSF topology is chosen here, as shown in Fig. 2. This topology has 16 nodes, and it is assumed that each node is also a datacenter. Three overall testcases are also defined and tested here. In particular, the first testcase is used to perform sensitivity analysis and fine tune the weighting factors to be used in the second and third testcases. Meanwhile, the second and third testcases are used to evaluate the proposed schemes for varying physical and links resource levels. Specifically, the second testcase, termed as an *under-resourced* scenario, uses lower values for both datacenters and connectivity resources. Meanwhile, the third testcase, termed as an *highly-resourced* scenario, emulates settings with higher resources levels.

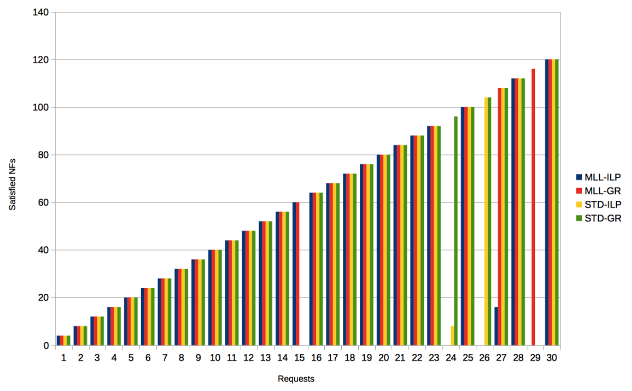
All testcases assume three different datacenter resources, i.e., W_d=$\{w_{d,1}, w_{d,2}, w_{d,3}\}$, representing processor, memory and storage. Namely, the first and second testcases set these values to $w_{d,1}$=$w_{d,2}$=$w_{d,3}$=500 units, respectively. Meanwhile, the third testcase sets all resource levels to 5,000 units. Similarly, in the first and second testcases, all links capacities are set to 1,000 units, for all $l_d^{i,j}$, where $(i, j)\in E$. Meanwhile, in the third testcase, these levels are increased to 10,000 units.

Finally, all other testcase values are defined as follows. First, the function set F has five types, 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 also uniformly distributed between $30 \leq w^i_{d,j} \leq 70$ units. Meanwhile the setup cost of placing an instance of function $i\in F_d$ at datacenter $d\in D$ is set to c^i_d=50. The instance capacity λ^i_d of a function i at datacenter d is also set to 2. It is also assumed that the set of functions F_d implemented by a datacenter d is comprised of all NFs $i\in F$. In other words, F_d=F. Finally, each request r requests four NFs, namely F_r is randomly selected from F ($F_r\subseteq F$).

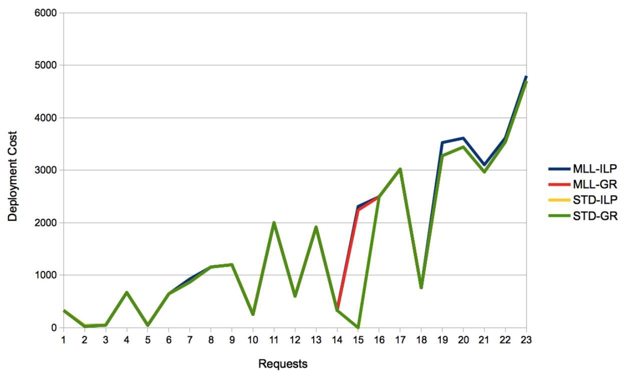
A. *Weighting Factors Selection*

Proper selection of the weighting factors for each term in the objective function, Eq. 1, is crucial for effective NF placement. Therefore the first challenge is to evaluate the impact (sensitivity) of these factors in order to maximize the number of satisfied requests, w_1, and minimize the deployment, routing and link load costs, i.e., w_2, w_3 and w_4, respectively.

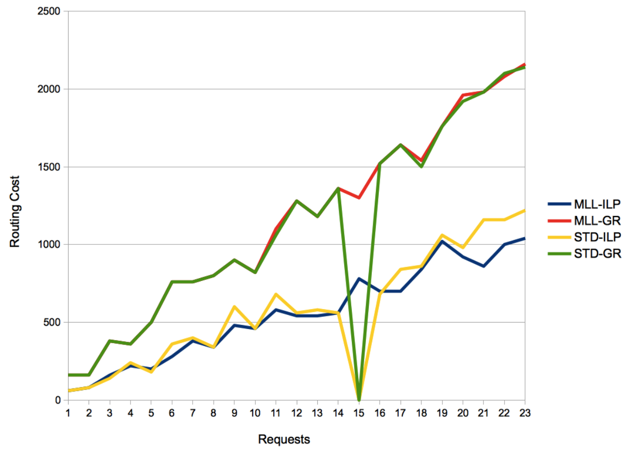
Now recall that the first term in Eq. 1 represents the number of satisfied NFs, which is typically a small value.



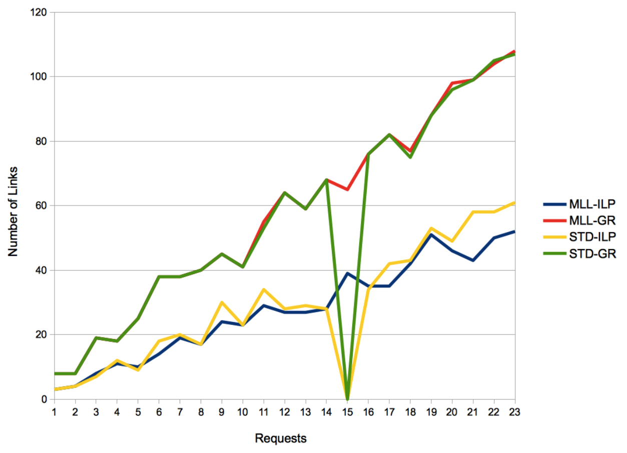
(a) Number of satisfied NFs



(b) Deployment cost



(c) Average deployment cost



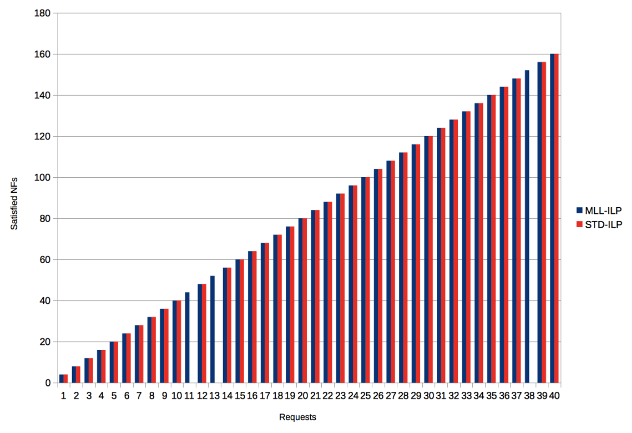
(d) Number of links

Fig. 4: Second testcase (under-resourced): request batch size 30

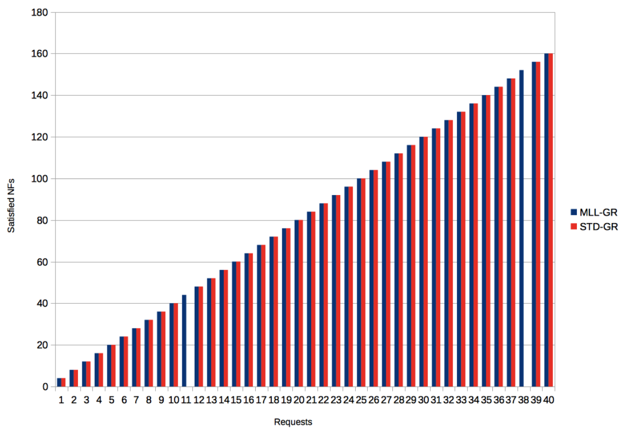
Meanwhile, the other terms represent costs with negative values (minimization). Hence it is imperative to assign a higher value to w_1. Therefore sensitivity analysis has to be done for the other weight values, i.e., w_2, w_3 and w_4. Now the deployment cost (second term) relates to the number of satisfied NFs i and their respective costs. Hence deploying multiple NFs in the same datacenter reduces the deployment cost of each instance capacity, λ_d^i, at a datacenter. On the other hand, the routing cost (third term) and the link load cost (fourth term) are associated with the number of links determined for each traffic flow. Although both of these costs are related to the number of hops, minimizing link load cost can result in longer paths. Clearly this trade-off needs to be studied further.

In general, deployment and routing costs are associated with a number of variables, i.e., up to $|D||F|$ and $|R||E|$, respectively. Hence increasing one of these minimization term factors minimizes the associated overall cost. However this approach may also increase the other costs. Accordingly, Figs. 3a and 3b show the trade-off between deployment and routing costs. Namely, for the first testcase, Fig. 3a compares the deployment cost for different values of w_2 and w_3. Overall, a larger routing cost weight (w_3=10) gives increased deployment costs, whereas a larger deployment cost weight (w_2=10) results in lower deployment cost. Next, Fig. 3b shows that higher routing cost weights result in lower routing cost and higher deployment cost. Finally, the link load minimization term (fourth term) only has a single variable α (ranging from 0 to 1). Therefore w_4 can be used as a balancing factor to select between one of the other two minimization terms, i.e., deployment or routing. Additionally, w_4 introduces increased sensitivity due to its decimal value (greater range). Note that this fourth term is also directly related to the third term, i.e., routing cost, since both use variables l_d^{ij} and b_r. Therefore minimizing α reduces routing cost and increases deployment cost. Furthermore, increasing w_4 decreases routing cost and increases deployment cost.

Next, to demonstrate the effect of varying w_4, an empirical methodology is deployed where two different values are used,



(a) MLL-ILP and STD-ILP



(b) MLL-GR and STD-GR

Fig. 5: Third testcase (highly-resourced): Number of satisfied NFs for request batch sizes from 1-40

i.e., w_4=10 and w_4=1,000. Tests are then performed in order to satisfy a number of requests, ranging from 1 to 20, i.e., 20 placement solutions with the values for w_1, w_2 and w_3 set to 1,000, 1 and 1, respectively. Here, Fig. 3c compares the average deployment cost for both values of w_4, i.e., computed as deployment cost/number of requests. Overall, these results show that the two costs are very similar. However Fig. 3d also shows that the average routing cost for w=10 is 2 to 4 times larger than the routing cost for w_4=1,000, i.e., computed as routing cost/number of requests.

Overall, the maximum link load variable α can be used to define the placement solution according to service provider needs. For example, some may prefer placing NFs to reduce deployment costs due to physical datacenter resource limitations. Meanwhile others may prefer to reduce routing cost due to network link transmission constraints, i.e., low link capacity, delay and management complexity, etc.

B. Under-Resourced Scenarios (Second Testcase)

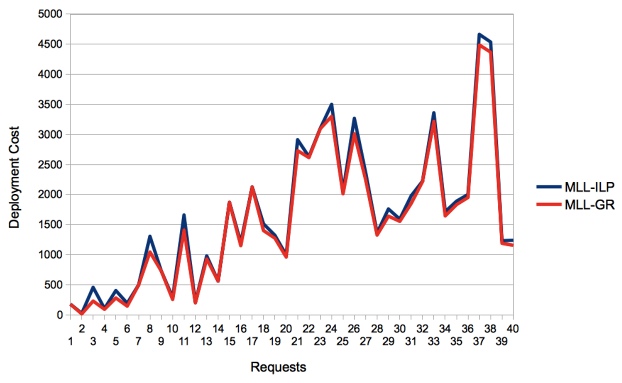
Based upon the above sensitivity analysis, the first three weighting factors are set to w_1=1,000 and w_2=w_3=1. All other parameters are defined as per the start of Section IV. Now the second (under-resourced) testcase assumes a total of 30 arriving requests. Fig. 4a plots the number of satisfied NFs for this scenario and shows that all four schemes begin to drop demands after 24 requests, i.e., due to resource limitations. However, note that both the STD-ILP and STD-GR schemes (yellow and green lines) fail to satisfy the 15th request due to their inability to minimize link loads. Specifically, these non-MLL schemes do not perform load balancing. Hence for subsequent plots, results are only shown for up to 23 requests.

Now the next plot in Fig. 4b compares the deployment cost. Clearly all schemes here yield very similar values (note STD-ILP and STD-GR drop to zero for request 15 since they cannot satisfy the requested NFs). It is also noted that from request 19 and onwards, the MLL-ILP scheme (blue) and STD-ILP scheme (yellow) introduce negligibly higher deployment cost (about 5%). However in terms of routing cost there is a huge gap between the ILP (blue and yellow) and greedy heuristic schemes (red and green), as shown in Fig. 4c. Namely, the greedy heuristics yield about twice as much routing cost than the ILP solutions. Clearly the latter are more efficient.

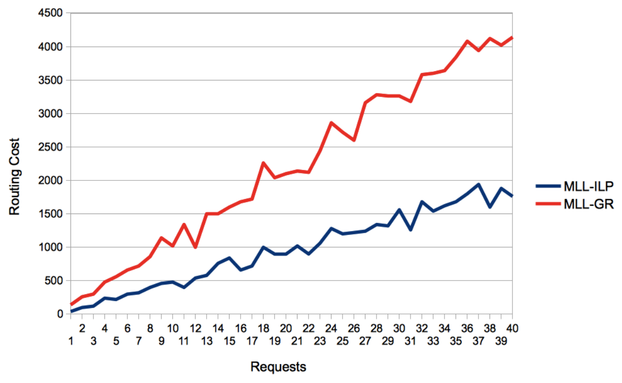
Note that the routing cost is directly related to the number of links used. Along these lines, Fig. 4d also plots the number of links associated with the deployed traffic flows. Again, there is notable similarity between the routing costs (about 100%). Additionally, for over 19 requests the MLL-ILP scheme starts to stabilize, whereas the STD-ILP scheme maintains its linear growth. Initially one may postulate that the MLL-ILP scheme should use more links than the STD-ILP scheme since it implements link load minimization. However, the MLL-ILP solution tries to minimize routing cost, which is directly related to link setup cost. Hence this scheme is more capable of minimizing routing cost and the number of links as well.

C. Highly-Resourced Scenarios (Third Testcase)

Most service providers deploy highly-resourced nodes in order to achieve rapid demand scalability. Along these lines the third testcase is designed to evaluate these scenarios. Foremost, initial tests are done to determine if all four schemes are appropriate for this type of network scenario, e.g., to differentiate between the MLL and STD schemes. Namely, Fig. 5a compares the two ILP schemes with regards to the number of satisfied NFs (for request batch sizes ranging from 1-40 requests). Here it is seen that the STD-ILP scheme fails to satisfy requests 11, 13 and 38 (unlike the MLL-ILP scheme). Equivalent results for the greedy heuristics in Fig. 5b also show that the STD-GR scheme fails to satisfy the above-noted requests as well (unlike the MLL-GR scheme). These findings clearly indicate that schemes which do not minimize link load exhibit higher blocking rates (by approximately 5 to 7.5%),



(a) MLL schemes deployment costs



(b) MLL schemes routing costs

Fig. 6: Third testcase (highly-resourced): MLL costs for request batch sizes from 1-40

even in such high-resourced testcases. In light of this, further tests are only done for the MLL-based schemes.

Fig. 6a plots overall deployment costs and confirms similar performance between the MLL-ILP and MLL-GR schemes. Furthermore both schemes can satisfy up to 160 requested NFs across 40 demands, i.e., 4 NFs per request. For example, the average overall deployment costs are 1,649.3 and 1,569.2 for the MLL-ILP and MLL-GR schemes, respectively (MLL-ILP deployment cost is 5% higher than MLL-GR). However, as Fig. 6b shows, the MLL-GR scheme gives much higher routing costs, i.e., approximately 128% higher (average overall routing costs are 982 and 2,221.5 for MLL-ILP and MLL-GR, respectively). Similarly, the MLL-GR heursitic also uses an average of 111 links to establish all traffic flows, whereas the MLL-ILP scheme only uses 48.6 links, as shown in Fig. 7. Finally, additional tests (not shown) are also done with larger batch sizes of up to 60 requests. Overall findings confirm successful placement of all NFs and the same relative performance between the MLL-ILP and MLL-GR schemes.

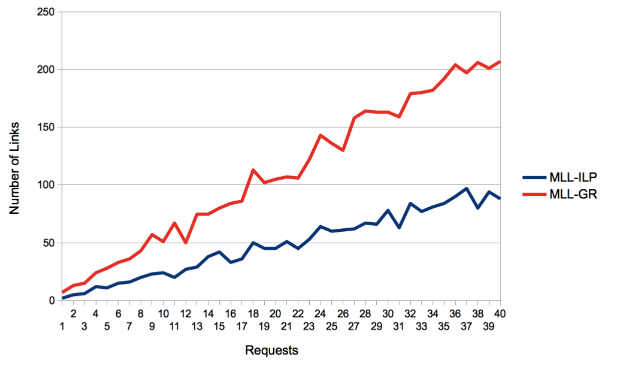


Fig. 7: Third testcase (highly-resourced): Link usage

V. Conclusions

Network function virtualzation (NFV) is a new technology that is helping reduce network management cost and complexity. Namely, this approach decouples network services from embedded hardware devices, enabling the execution of network functions as software instances on commercial-of-the-shelf (COTS) systems. However, NFV-based deployments pose new challenges of their own, including the network function (NF) placement problem. As a result, various placement schemes have been proposed recently. However, most of these studies focus on minimizing cost and assume that there are adequate resources to satisfy all requests. Although some efforts have looked at maximizing the number of satisfied NFs, they have not considered link capacity constraints and link load balancing. To address these concerns, this paper presentes a novel integer linear programming (ILP) solution to jointly route and place NFs, termed as the *minimized link load ILP* (MLL-ILP) scheme. This strategy guarantees that a traffic flow designated to satisfy a request has adequate bandwidth capacity, and it also minimizes link load. The MLL-ILP solution also includes further provisions to allow service providers to select between deployment and routing costs. However, owing to ILP scalability concerns for larger networks, a greedy heuristic algorithm is also proposed. Both strategies are evaluated for a range of network testcases, including under-resourced and highly-resourced scenarios. Overall findings confirm that the optimization-based MLL-ILP approach gives notable routing improvements over its heuristic counterpart for both scenarios. However the deployment costs are very similar.

References

[1] Mijumbi, R., Serrat, J., Gorricho, J., Bouten, N., De Turck, F., Boutaba, R., "Network Function Virtualization: State-of-the-Art and Research Challenges," *IEEE Communications Surveys & Tutorials*, Vol. 8 (1), 2016.

[2] Kreutz, D., Ramos, F., Verissimo, P., Rothenberg, C., Azodolmolky, S., Uhlig, S., "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, Vol. 103 (1), January 2015, pp. 14-76.

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

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

[5] Xia, M., Shirazipour, M., Zhang, Y., Green, H., Takacs, A., "Network Function Placement for NFV Chaining in Packet/Optimal Datacenters," *IEEE/OSA Journal of Lightwave Technology*, Vol. 33, Issue 8, April 2015, pp. 1565-1570.

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

[7] Crichigno, J., Oliveira, D., Pourvali, M., Ghani, N., Torres, D., "A Routing and Placement Scheme for Network Function Virtualization," *International Conference on Telecom. and Signal Processing (TSP) 2017*, Barcelona, Spain, July 2017.

[8] Crichigno, J., Ghani, N., Khoury, J., Shu, W., Wu, M., "Dynamic Routing Optimization in WDM networks," *IEEE Global Communications Conference (GLOBECOM) 2010*, Miami, FL, December 2010.

[9] Crichigno, J., Shu, W., Wu, M., "Throughput Optimization and Traffic Engineering in WDM Networks Considering Multiple Metrics," *IEEE International Conference on Communications (ICC) 2010*, Cape Town, South Africa, May 2010.

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

Diogo Oliveira is a Research Assistant in the Dept. of Electrical Engineering at the University of South Florida (USF), where he is working towards his Ph.D. degree. He received his M.Sc. degree from the Federal University of Goias (UFG), Brazil, in 2009. His current research interests include sofware defined networking (SDN), network virtualization and services, disaster recovery, and design and application of optimization and meta-heuristic algorithms.

Jorge Crichigno received his Ph.D. in Electrical and Computer Engineering from the University of New Mexico, Albuquerque, NM, in 2009. Prior to that, he received his M.Sc. and B.Sc. in Electrical Engineering from the University of New Mexico and from the Catholic University of Asuncion respectively, in 2008 and 2004. Dr. Crichigno is currently an Associate Professor in the Department of Integrated Information Technology in the College of Engineering and Computing at the University of South Carolina, Columbia, SC. In 2016, he was a visiting professor in the Florida Center for Cybersecurity, Tampa, FL. His research interests include wireless and high-speed networks, network security and Science DMZs, and STEM education. He has served as reviewer and TPC member for journals and conferences such as IEEE Transactions on Mobile Computing and IEEE Globecom, and as panelist for NSF STEM education initiatives. He is a member of the IEEE Computer Society.

Nasir Ghani is a Professor in the Electrical Engineering Department at the University of South Florida and Research Liaison for the Florida Center for Cybersecurity (FC2). Earlier he was Associate Chair of the ECE Department at the University of New Mexico. He has also spent several years working at large Blue Chip organizations (IBM, Motorola, Nokia) and hi-tech startups. His research interests include cyberinfrastructure networks, cybersecurity, cloud computing, and cyber-physical systems. He has published over 200 articles, and his research has been supported by many organizations. He is also the recipient of the NSF CAREER Award and is an Associate Editor for IEEE/OSA Journal of Optical and Communications and Networking. He has also served on the editorial boards of IEEE Systems and IEEE Communications Letters. He has co-chaired many symposia for IEEE GLOBECOM, IEEE ICC, and IEEE ICCCN. He received the Ph.D. from the University of Waterloo.editorial boards of IEEE Systems and IEEE Communications Letters. He has co-chaired many symposia for IEEE GLOBECOM, IEEE ICC, and IEEE ICCCN. He received the Ph.D. from the University of Waterloo.