Real Traffic-Aware Scheduling of Computing Resources in Cloud-RAN

Hatem Khedher, Sahar Hoteit, Patrick Brown, Véronique Vèque, Ruby Krishnaswamy, William Diego, Makhlouf Hadji

Dynamic Placement of Extended Service Function Chains: Steiner-based Approximation Algorithms

Selma Khebbache
Institut Mines-Telecom
Telecom SudParis
UMR 5157, Samovar
selma.khebbache@telecom-sudparis.eu

Makhlouf Hadji
Technological Research
Institute SystemX
Palaiseau, France
makhlouf.hadji@irt-systemx.fr

Djamal Zeghlache
Institut Mines-Telecom
Telecom SudParis
UMR 5157, Samovar
djamal.zeghlache@telecom-sudparis.eu

***Abstract*—This paper proposes Steiner-based algorithms to extend already deployed tenant slices or Virtualized Network Functions Forwarding Graphs (or Service Function Chains) as demand grows or additional services are appended to prior service functions and chains. The tenant slices are hosted by Network Function Virtualization Infrastructure (NVFI) providers that can make use of the proposed algorithms to extend tenant slices on demand for growing traffic loads and service extensions including protection and security services(such as extending a slice with a dedicated security slice). The paper proposes a Steiner-based ILP as an exact solution for small graphs and Steiner based approximation algorithms to improve scalability for larger problems.**

I. Introduction

Network Function Virtualization addressed by ETSI [3] has the potential of changing the way future shared and virtualized infrastructures are used to support service providers. NFV enables provisioning of on demand networking services, offers connectivity as a service and eases development of network services and their management [7]. NFV implements networking functions as software that can be run on a variety of physical hosts (see [1] and [3]). By decoupling network services and functions from the hardware, NFV enables service providers or tenants to request on a need basis hosting resources to physical infrastructure providers. In NFV the providers can acquire Virtualized Network Functions (VNFs) from third party and combine them with their own VNFs to build dynamically their dedicated virtual infrastructures and deploy the resulting service graphs in the hosting platforms offered and managed by the infrastructure providers. The service providers will build their infrastructure gradually as consumer and customer demand grow and hence will request additional resources and services to extend their already deployed services. The service providers (or tenants) may add new VNFs, that the providers will need to host and connect to previously deployed and activated tenant services, may insert new VNFs in existing forwarding paths, may remove, replace or migrate VNFs dynamically.

This foreseen usage of NFV Infrastructure services by tenants or service providers put new optimization and hosting requirements to the infrastructure providers. The current state of the art is mainly concerned by optimal initial embedding and placement of virtual networks and services in provider networks and their dynamic adaptation for varying demand and traffic conditions. The problem of extending an already deployed tenant slices has not been addressed at large. In previous works dynamic changes are minor, local or very basic extensions of deployed service graphs such as spawning new VNFs, up and down scaling of hosts and containers. Adding complex service graphs to already deployed slices has not been considered to the extent presented in this paper that focusses the study on the extension of already deployed service function chains (SFC in IETF terms) in shared infrastructures.The solution proposed in this work is applicable not only to extensions of already running service graphs but also to insertions in and modification of service graphs. The problem is hence approached by searching for an optimal placement, of complex service chains extensions in physical infrastructures, that meets the requested connectivity with the previously deployed service graph and that does not disrupt the initial deployment. Our contribution consists in proposing an Integer Linear Programming Steiner-based approach and two new approximation algorithms based on the construction of a Steiner tree that covers a large part of the convex hull of the considered Extended-SFC (E-SFC) placement problem. To the best of our knowledge this is the first time the problem of extending already deployed complex SFCs is addressed and treated using Steiner-based algorithms.

Section II of this paper presents related work on SFC placement that usually takes only into account placement constraints and addresses scalability of the algorithms and typically does not consider extension requests of already deployed service chains. Section III is dedicated to the SFC extension problem description and formulation. New approximation algorithms based on a smart Steiner-tree construction, to improve scalability and find solutions in practical times, are also presented in section III. Section IV reports performance evaluation results.

II. Related Work

Authors of [5] address the placement problem of SFCs or VNF-FG using approximation solutions to realize near-

optimal simultaneous placement of the VNFs and the chains. An exact formulation is provided by a 2-Factor modeling of the placement problem. Two approximation solutions based on a matrix approach and a multi-stage algorithm are also presented. The algorithms address only initial placement of the tenant requested service function chains. The extension of already deployed graphs due to new requests and dynamic variations is not considered while this is our central objective. Authors of [2] propose an orchestrator framework including a resource monitoring system for automated placement of VNFs and VNF chains. The framework manages the creation, configuration, instantiation, activation and removal of the VNFs but does not address the extension of operational SFCs requested by tenants to evolve their services and respond to increasing demand. Our objective is to propose algorithms to extend already running SFCs with new network forwarding paths and new service graphs. This type of extension is harder to achieve since the previous services and service chains should not be affected and the connectivity with the requested extensions established while respecting old and new requirements and constraints.

In our case, we build our placement and extension algorithms using a Steiner-tree representation, construction and resolution of the problem. We consequently provide the needed background on Steiner trees and their relationship to graphs. Reference [6] is recommended reading for the minimum Steiner tree problem. Our work relies on this background to address the service graph extension problem in the NFV context.

III. System Model and Problem Formulation

Figure 1 depicts a typical scenario for the optimal placement of SFC extension requests composed of a chain with two VNFs f_4 and f_5 strongly connected to the previously deployed SFC composed of f_1, f_2 and f_3. The new chain (extended request) starts from f_1 (already deployed on server S_3) and ends in VNF f_3 (already deployed on server S_5).

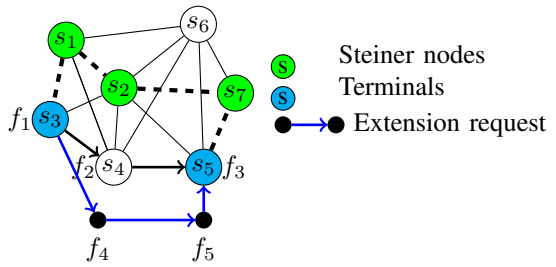


Fig. 1: A Steiner-based tree for E-SFC placement

For this scenario, the objective is to extend the already deployed VNF chain $f_1 \rightarrow f_2 \rightarrow f_3$ with a new chain $f_1 \rightarrow f_4 \rightarrow f_5 \rightarrow f_3$. The solution must ensure that the new VNFs f_4 and f_5 , part of the graph extension request, are connected to the previously deployed VNFs f_1 and f_3, respectively. The optimal placement of f_4 and f_5 has to comply with this connectivity obligation, namely $f_1 \rightarrow f_4$ and $f_5 \rightarrow f_3$ as depicted in Figure 1. The objective is to find the best hosts for f_4 and f_5 and the optimal path connecting them while ensuring that the best links to the previously deployed VNFs f_1 and f_3 are also provided by the algorithm.

A. *Steiner trees background and construction*

Definition The addressed graphs are represented by an undirected graph $G = (V, E)$ with a finite set of vertices V, and set E of associated edges. Each edge e has a weight $w_e \in \mathcal{R}^+$. In the optimal Steiner tree problem, we start from a set $X \subseteq V$ of special vertices and search for minimum cost tree $T \subseteq G$ connecting all the vertices in X with extra intermediate vertices and edges. The vertices in X are called **Terminals** and the vertices in $V(T) \backslash X$ are **Steiner vertices**.

The E-SFC extension can be cast into the previous definition of the Steiner tree problem by considering $X = \{S_3, S_5\}$, the hosts of VNFs f_1 and f_3 respectively in Figure 1, as terminals, and $V(T) \setminus X = \{S_1, S_2, S_7\}$ as the Steiner vertices. This representation shows that the E-SFC problem can be seen as a minimum cost Steiner tree problem.

B. *Mathematical formulations and modeling*

*1) **Exact formulation**:* The proposed exact mathematical formulation adapts the construction of the Steiner tree to the E-SFC problem. In the model, the hosting physical infrastructure is represented by a graph $G = (V, E)$ with a set V of vertices (physical servers) and a set E of edges. Each edge e has a nonnegative weight w_e that represents the amount of bandwidth *available* on the edge e. Our objective is to construct a minimum cost (in terms of total weight of selected edges) that covers a subset of nodes or vertices noted by T and corresponding to the notion of *terminals* in the Steiner tree problem. We accordingly construct a tree covering the identified terminal nodes T, when selecting a subset of other nodes S different from T, that is $S \neq T$. The set S of selected nodes correspond to *Steiner* nodes.

The exact formulation consists of an objective function that minimizes the total cost of the Steiner tree. In Figure 1, we express as req the amount of flow requested by the SFC extension and check if this amount is lower then the available bandwidth on edge (link) e using the following function:

$$\mathbb{1}_e = \begin{cases} 1, & \text{if } w_e - req \geq 0; \\ 0, & \text{else.} \end{cases} \tag{1}$$

For each edge $e \in E$, we introduce a binary variable x_e to indicate if edge e is in the Steiner tree ($x_e = 1$) or not ($x_e = 0$). Based on these notations, the objective function to minimize is expressed as follows:

$$\min Z = \sum_{e \in E} (w_e - req)\, \mathbb{1}_e x_e \tag{2}$$

This objective function is combined with valid inequalities and constraints to ensure that nodes selected for hosting will be part of the Steiner tree resolution.

To simplify the notations for the used constraints by the Branch-and-Cut algorithm, we introduce for each set of

nodes X, a set $\delta(X)$ composed of *edges* with one end node in X and the other one in the complement of X (i.e. $\bar{X}$). The first constraint that should be taken into account by the ILP is:

$$x\left(\delta(W)\right) \geq 1, \forall W \subseteq V, W \cap T \neq \emptyset, (V \setminus W) \cap T \neq \emptyset \quad (3)$$

The role of this constraint is to push the objective function towards a Steiner tree of minimum weight as a function of the identified terminal nodes. The objective function actually eliminates edges that do not fulfill the SFC extension request while the constraint steers the solution to Steiner tree that covers the desired terminal nodes or set T.

The final model of the ILP using a Branch-and-Cut approach is summarized as:

$$\begin{array}{l} \min Z = \sum_{e \in E} \left(w_e - req\right) \mathbb{1}_e x_e \\ S.T.: \\ \begin{cases} x\left(\delta(W)\right) \geq 1, & \forall W \subseteq V, W \cap T \neq \emptyset, (V \setminus W) \cap T \neq \emptyset \ ; \\ x_e \in \{0;1\}, & \forall e \in E; \end{cases} \end{array} \quad (4)$$

*2) **Approximation algorithms**:* We propose two heuristic approaches (Shortest Path-based Heuristic and Minimum Spanning Tree-based Heuristic) described below.

Algorithm 1 SP-Heuristic algorithm

Input: A weighted graph G (physical infrastructure after removing edges that do not satisfy the request), and a set of Terminals T
Initialization: Steiner-tree = $\emptyset$
Choose an arbitrary node r in T
while $T \neq \emptyset$ **do**
  Select a node $s \in T$
  Calculate the Shortest Path P_{sr} from s to r
  Steiner-tree $+= P_{sr}$
  $T = T \setminus s$
end while
Check the chaining feasibility on the obtained Steiner tree

Algorithm 2 MST-Heuristic algorithm

Input: A weighted graph G (physical infrastructure after removing edges that do not satisfy the request), and a set of Terminals T
Calculate a minimum spanning tree SP_{tree} of G
Using the previous tree SP_{tree}, delete all of the leaves that are **not terminals** in T

IV. Numerical Results

The performance evaluation of the algorithms, is conducted using a 2.40 GHz PC with 8 GB RAM. Each SFC comprises a random number of VNFs in the $[2;10]$ interval. The physical hosts have random number of available CPUs drawn from a $[100;150]$ CPUs range, while each VNF in the SFC has a required processing capability in the $[1,20]$ CPU range. The traffic loads for the requested VNFs (expected traffic flow across the VNFs) in the SFCs are drawn randomly in the $[1,5]$ Mbps range. The simulation duration is 100000 time units for each run. SFC Extension requests arrive following a Poisson process with an average arrival rate λ of 1 arrival per 120 time units and have an exponential service rate μ of 1 departure every 350 time units, corresponding to a rather heavy system load condition.

The simulation and experiments use CPLEX as an optimization solver for the Steiner-based ILP (4), and assess the performance of our algorithms using a real topology from the library of test instances for Survivable fixed telecommunication Network Design [4].

Table I reports the performance of the approximate algorithms compared with the exact ILP formulation. The metrics used for this comparison are:

- **Convergence time**: is the time needed by the algorithms to converge to their best solutions to the E-SFC problem.
- **Gap**: is used to assess the relative performance (cost) of the approximation algorithms compared with the performance (cost) of the exact ILP formulation algorithm used as "the reference and optimal solution". This metric reflects the relative quality of the approximate algorithms compared with the exact ILP achieved cost and this is given by $Gap_1(\%) = \frac{Z^*_{SP} - Z^*_{ILP}}{Z^*_{ILP}} \times 100$ ($Gap_2(\%) = \frac{Z^*_{MST} - Z^*_{ILP}}{Z^*_{ILP}} \times 100$, respectively) representing the gap between the best solution of the SP method (Z^*_{SP}) (MST method with a best solution Z^*_{MST}, respectively) compared with the optimum Z^*_{ILP};

TABLE I: Exact vs Approximation Algorithms Performance

# Servers	SP Time (s)	MST Time (s)	ILP Time (s)	SP $Gap_1(\%)$	MST $Gap_2(\%)$
5	<0.01	<0.01	0.11	0.17	0.15
10	<0.01	<0.01	0.11	3.20	5.80
30	<0.01	<0.01	0.20	3.50	5.90
50	0.02	0.03	0.22	3.70	6.10
80	0.13	0.12	0.59	3.80	6.80
100	0.23	0.27	1.44	3.90	6.80
130	0.67	0.73	1.57	3.90	6.74
150	1.07	0.78	2.19	3.93	8.10
180	1.96	1.38	17.21	6.81	9.30
200	2.54	3.04	> 3mn	9.76	10.13

Table I, that reports the convergence time performance results for the ILP and the approximation algorithms, highlights the efficiency of the SP and MST algorithms in finding solutions much faster than the Exact ILP algorithm. They also scale much better than the ILP. The SP algorithm needs $2.54s$ for infrastructures with 200 nodes while the MST requires slightly longer times as problem size increases. Both have much closer performance for smaller graphs. The exact ILP formulation, that explores all solutions, and finds the optimum can be used for graphs of around a hundred nodes (still usable up to 180 nodes but convergence time increases to tens of seconds, around 18 s). This is expected since the E-SFC extension problem is NP-Hard. Larger problem sizes require the use of the proposed Steiner based heuristics that can find solutions in a few seconds (2.54 s

for SP and 3.04 s for 200 physical nodes, servers or hosts).

The convergence time performance of the algorithms has to be taken into account jointly with other metrics, especially with the cost of the solution assessed as gap between the approximation algorithms and the exact ILP solution as well as with the proportion of E-SFC requests that are rejected in the simulation runs. The gap between the approximation algorithms and the ILP increases with problem size (increasing number of hosts) as reported in the two right most columns of Table I. The gap for small sizes is negligible for example for 5 servers, it is as low as 0.17% and 0.15% respectively for the SP and MST heuristics. The gap increases however beyond 5% above 10 servers and can reach around 10% for hundreds of candidate hosts.

The exact ILP solution outperforms the approximation solutions (SP and MST) in rejection rate since the ILP explores all the space and finds the optimal paths for the E-SFC extension request if enough physical resources are available. Figure 2 depicts the rejection rate simulation results for increasing infrastructure sizes. As the load decreases, the rejection rate drops from the maximum of 1.15% for 5 hosts to zero (0%) for more than 80 servers. For this simulated operating zone, the SP and MST reject 4% (for 50 hosts or less) and 3% (when the number of servers is in the range 50 to 200 hosts) which can be considered quite acceptable. These results show that the approximation algorithms can provide good solutions much faster than the ILP while not sacrificing that much rejection rate and quality of the solutions. ThE SP and MST are equivalent in terms of rejection rate.

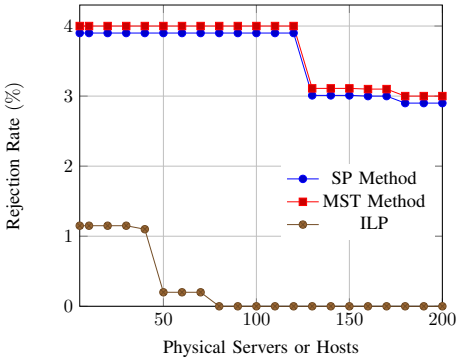
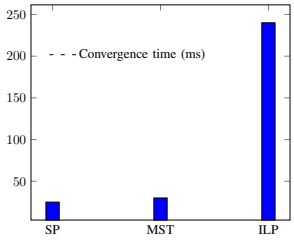
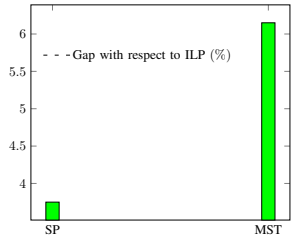


Fig. 2: Rejection rate comparison

To extend the analysis and performance assessment using random graph generations for the proposed algorithms, a real European topology (with 37 nodes and 57 edges) is used to complete the study. Figures 3a and 3b report the proposed algorithms performance results for this European network topology [4]. Figure 3a reveals for this topology, small differences in convergence time between the SP and MST algorithms ($\sim$ 20 ms for SP and $\sim$ 25 ms for MST to find feasible solutions) while the ILP requires an order of magnitude more time to find the optimal solution (250 ms). This real network instance is in fact "easy" to solve using our approximation algorithms. Even the exact approach based on



(a) Convergence time on a real toplogy



(b) SP and MST gap comparison on a real topology

the Branch-and-Cut method manages to find the optimal in 250 ms.

Note, however, that for more complex extensions (containing multiple forwarding paths and dense neworks with high connectivty), the MST will outperform the SP in convergence time and will improve in performance with gaps closer to the SP algorithm. The simulations results reported in Figures 3a and 3b are more favorable to the SP, since the extensions contain typically few paths and VNFs and weakly connected hosting infrastructures. When multiple paths are involved, the SP will be called multiple times and will take longer to find solutions whereas the MST will provide in a single run the solution much faster.

V. Conclusion

This paper proposes Steiner-based algorithms to extend already deployed tenant slices with new virtualized network functions organized in complex chains and service graphs. An exact ILP algorithm based on the description of the convex hull of the extension problem, appropriate for small and medium size instances, is compared with two proposed Steiner based approximation algorithms (Shortest Path and Minimum Spanning Tree algorithms). Simulations based on randomly generated graphs and performance evaluations using real topologies and traces show that the proposed Steiner based approximation algorithms find good solutions (close to the ILP) much faster than the ILP and scale much better with problem size. To the best of our knowledge this is the first time the extension of slices (service function chains or VNF forwarding graphs) is addressed with minimum Steiner trees approaches for their near optimal placement.

References

[1] "A comprehensive survey of network function virtualization," *Computer Networks*, vol. 133, pp. 212 – 262, 2018.

[2] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The dynamic placement of virtual network functions," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–9.

[3] ETSI, "Network functions virtualization (nfv); architectural framework."

[4] http://sndlib.zib.de/home.action, 2018.

[5] S. Khebbache, M. Hadji, and D. Zeghlache, "Virtualized network functions chaining and routing algorithms," *Computer Networks*, vol. 114, pp. 95–110, 2017.

[6] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*.

[7] ONF, "Software-defined-networking: The new norm of networks, onf white paper."