Advanced Topics on Cloud Computing **Spring 2021**

Lecture 4: Network Slicing

Lecturer: Dr. Georgios Paschos *Amazon*

In cloud computing we have pools of computing, memory, and network resources and we desire to allocate them to users in order to build applications. A generic way to allocate resources is by designing virtual networks, called *slices*, which are built by resource units from the available pools. For example, network functionality like firewalls, caches, and authentication can be performed by software middleboxes, called *Virtual Network Functions* (VNFs). The VNFs can run at different cloud locations on general purpose computing hardware, and traffic can be steered among them remotely using *Software Defined Networks* (SDN). Combining these two enabling technologies, VNFs and SDN, it becomes possible to launch applications *when-and-where*, satisfying diverse and demanding requirements, in the form of network slices [23]. Examples of such applications include but are not limited to: (i) control of autonomous vehicles, (ii) virtual/augmented reality and remote surgery, (iii) high accuracy industry communications, and (iv) control of robots and smart grids. The contribution of network slicing to these applications is that it will allow them to co-exist on the same infrastructure. However, to effectively allocate resources to these different slices, we require a nimble controller that continuously solves resource allocation problems and quickly decides what is the appropriate embedding of the virtual network.

This writeup provides the definition of the Virtual Network Embedding problem, which can be used to determine the optimal resource allocation for virtual networks. We also provide a proof of the complexity of this problem in the case of splittable flows.

4.1 Virtual Network Embedding

In the problem of *Virtual Network Embedding* (VNE) with unsplittable flows (termed hereinafter VNE-UF) we are given a physical network with costs and capacities on links and we are asked to find an embedding of a virtual network onto the physical with minimum cost [11]. The virtual network consists of a graph with virtual nodes (vnodes) and virtual links (vlinks), and its embedding consists in assigning each vnode to a physical node among a set of options, and each vlink to a path in the physical network connecting the corresponding vnode embeddings, see Fig. 4.1. In cloud computing, the physical network plays the role of available resource pools, and the virtual network plays the role of (i) a user application, or (ii) a nested virtual infrastructure.

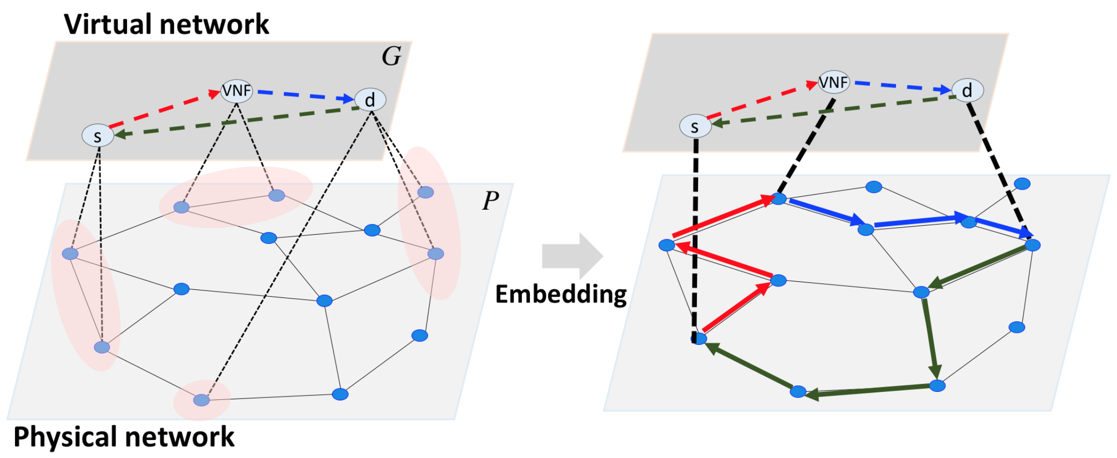

Figure 4.1: Virtual network embedding with unsplittable flow.

Physical network: Our network infrastructure is described by a graph $P = (N, L, \boldsymbol{b}, \boldsymbol{c})$ with physical nodes N and links L. The total amount of flow that can be routed through link $l \in L$ is limited by capacity b_l, and there is a routing cost c_l for each unit of flow. Depending on the application, the costs may be monetary (e.g. when renting resources), or refer to metrics such as energy and congestion.

Virtual network: We receive requests for virtual networks to be implemented on the physical infrastructure. A request is depicted by another graph $G = (V, E, \boldsymbol{M}, \boldsymbol{d})$, with virtual nodes (vnodes) V and virtual links (vlinks) E. Each vnode $v \in V$ has a set of options $M_v \subseteq N$, i.e., physical nodes where it can be embedded. Each vlink $e \in E$ is associated to a demand d_e, which denotes the total amount of flow that will be circulated on the virtual network over this vlink.

Virtual network embedding for unsplittable flow: The embedding of G on P in the unsplittable flow scenario asks for the embedding of vnodes V and the embedding of vlinks E. To *embed vnode* v we simply need to pick one physical node from the options M_v. This can be achieved by the use of binary variables $x_{vn} \in \{0, 1\}$ that take value 1 iff the vnode v is embedded on node n. To *embed vlink* $e = (v_1, v_2)$ we need to find an acyclic path in the physical network P that connects the embeddings of v_1, v_2, i.e. that connects two physical nodes n_1, n_2 such that $x_{v_1 n_1} = x_{v_2 n_2} = 1$. Let $y_{el} \in \{0, 1\}$ take value 1 to denote that vlink e is embedded on a path that includes link l and 0 otherwise. For each vlink $e = (v_1, v_2)$, ensuring that the link embedding takes the form of a path on the physical network corresponds to certain flow-type constraints at each physical node:

$$\sum_{j \in Out(n)} y_{e(n,j)} - \sum_{j \in In(n)} y_{e(j,n)} = x_{v_1 n} - x_{v_2 n}, \quad \forall n \in N,$$

where $In(.)$ and $Out(.)$ denote the set of incoming and outgoing neighbor nodes respectively. Observe that the term $x_{v_1 n} - x_{v_2 n}$ can take values $1, 0, -1$ depending on whether the inspected node n is a source, an intermediate, or a destination node of the path, as is customary with flow conservations.

The objective is to find a feasible virtual network embedding that minimizes the total cost. This leads to the problem of Minimum Cost *Virtual Network Embedding with Unsplittable Flow* (VNE-UF), formalized in (4.1)-(4.4).

<u>VNE-UF:</u>

$$\underset{\substack{\boldsymbol{x} \in \{0,1\}^{VN} \\ \boldsymbol{y} \in \{0,1\}^{EL}}}{\text{minimize}} \quad \sum_{\substack{e \in E \\ l \in L}} c_l d_e y_{el} \tag{4.1}$$

$$\text{subject to} \quad \sum_{e \in E} d_e y_{el} \le b_l, \ \forall l \in L, \tag{4.2}$$

$$\sum_{n \in N} x_{vn} = 1, \ \ x_{vn} = 0, \text{ if } n \notin M_v, \ \forall v \in V, \tag{4.3}$$

$$\sum_j y_{e(n,j)} - \sum_j y_{e(j,n)} = x_{v_1 n} - x_{v_2 n}, \forall n \in N. \tag{4.4}$$

The objective $\sum_{e \in E, l \in L} c_l d_e y_{el}$ reflects the total cost of physical link usage by our vlink embeddings. Constraint (4.2) is the link capacity constraint, constraint (4.3) ensures that vnode v is embedded only once and only at available options M_v, while constraint (4.4) is the flow conservation mentioned above.

The VNE-UF is very general as it subsumes a number of important problems such as (i) the resource allocation for network slices [22] (ii) the VNF chaining problem [3], (iii) the joing optimization of routing and VNF placement [1], (iv) the optimal functional split in C-RAN [18], (v) the embedding of computation graphs on networks [24], and (v) the minimum cost multicommodity flow problem [9]. In extended variants, it might be useful to consider (i) multiple virtual networks [4], (ii) QoS and survivability constraints [22], as well as (iii) online variants [7, 25].

The VNE-UF problem is an Integer Linear Program. Fixing the node embedding, i.e., assuming M_v sets are all singletons, the problem becomes the well-known minimum cost unsplittable multicommodity flow, which is known to be *NP*–hard [9]. In [21] it was shown that VNE-UF is APX-hard, i.e. it is *NP*–hard to approximate within a constant factor.

4.2 Virtual Network Embedding with Splittable Flows

In general, we would like to simplify the VNE-UF problem as much as possible such that its solution is still useful. To this end, note that *splitting the flow* is possible in many modern SDN systems [26]. Hence, in this paper we

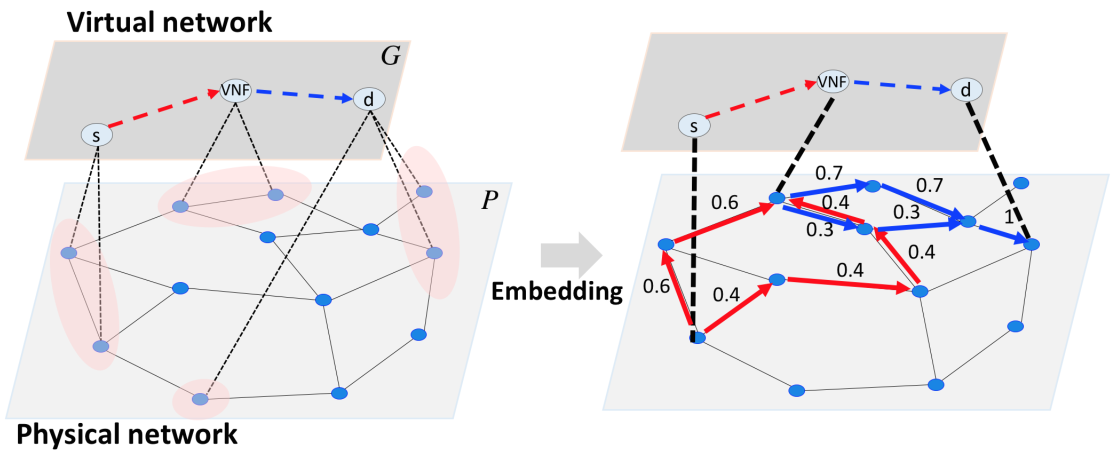

Figure 4.2: An example of a virtual network embedding with splittable flow. The vlink demands are split over multiple paths, which constitute a splittable flow.

consider the VNE problem with splittable flows (VNE-SF), where instead of paths, the vlinks are embedded with the form of a splittable flow, i.e., the vlink demand can be transported over multiple paths. We remark that *splitting the VNF placement* over multiple computing nodes is also possible by means of resource disaggregation [16, 5], but it creates the overhead of maintaining the state consistency between the different parts of the VNF, and hence we leave this consideration out of scope. We may observe that VNE-SF is significantly easier than VNE-UF: if we fix the node embedding as before, the remaining problem is a Linear Program (LP) solvable in polynomial time. Below, we show that the VNE-SF remains *NP*–hard emplying a reduction from the 3-SAT problem.

4.2.1 VNE for Splittable Flow

We relax the integrality of the vlink embedding variables y_{el} and allow the embedded vlinks to circulate traffic in the form of a splittable flow, which flows over a set of paths instead of a single path. This can be done by simply letting $y_{el} \in [0,1]$, in which case y_{el} denotes the fraction of vlink demand d_e which is routed over physical link l. The corresponding problem becomes:

VNE-SF:

$$\underset{\substack{\boldsymbol{x}\in\{0,1\}^{VN}\\ \boldsymbol{y}\in[0,1]^{EL}}}{\text{minimize}} \quad \sum_{\substack{e\in E\\ l\in L}} c_l d_e y_{el} \tag{4.5}$$

$$\text{subject to} \quad \sum_{e\in E} d_e y_{el} \le b_l,\ \forall l\in L, \tag{4.6}$$

$$\sum_{n\in N} x_{vn} = 1,\ \ x_{vn} = 0,\ \text{if } n\notin M_v,\ \forall v\in V, \tag{4.7}$$

$$\sum_j y_{e(n,j)} - \sum_j y_{e(j,n)} = x_{v_1 n} - x_{v_2 n}, \forall n\in N. \tag{4.8}$$

4.2.2 The Multipartite Graph

Fundamental to our approach is the construction of the *multiparite graph* $\hat{G} = (\hat{V}, \hat{E}, \hat{\boldsymbol{c}})$, which aims to capture the combinatorial structure of embedding options. The nodes $\hat{V}$ are defined to be the union[1] of all candidate physical nodes $\hat{V} = \cup_{v\in V} M_v$, and the links $\hat{E}$ are defined in the following way. First, for each vlink $e = (v_1, v_2) \in E$ we

[1]Taking the union as the naming convention works only for disjoint sets M_v (but simplifies exposition). In case they are not disjoint, the multipartite graph can be formulated with alternative naming convention such that each physical node appears as many times as in the sets M_v.

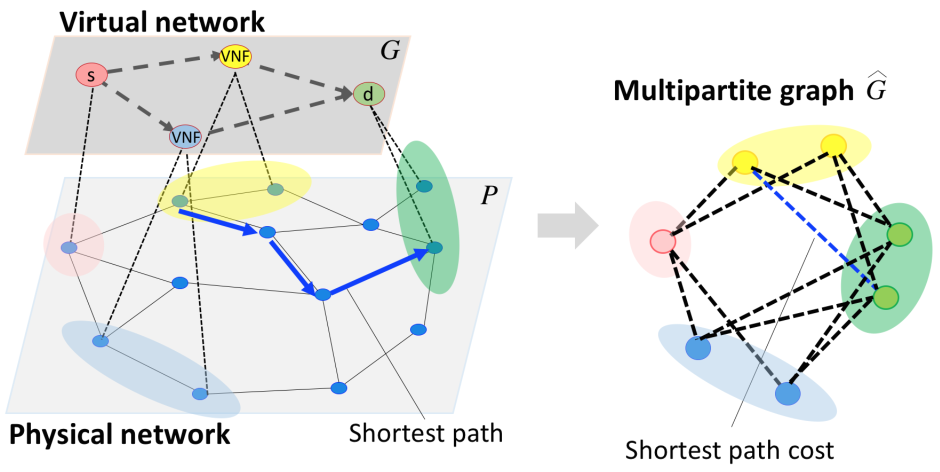

Figure 4.3: An example of a multipartite graph construction.

identify the two node subsets M_{v_1}, M_{v_2} and then we denote with $E_R(l)$ the links of the utility graph formed by these two subsets (i.e. we connect all nodes of M_{v_1} to all nodes of M_{v_2}. Ultimately, we define $E_R = \cup_{e \in E} E_R(l)$. Last, on each link $e = (n_1, n_2) \in \hat{E}$ we introduce the cost $\hat{c}_e$ which equals to the shortest path cost connecting the two physical nodes $n_1, n_2 \in N$. We provide in fig. 4.3 a pictorial example.

We note that the multipartite graph can be constructed in polynomial time by computing all involved shortest paths, which can be done in, e.g., $O(|N|^3)$ time via Floyd-Warshall.

Recall that x_{vn} is 1 iff vnode v is embedded on physical node n. Consider the following subgraph selection problem, which we term the multipartite graph embedding.

MGE:

$$\underset{\boldsymbol{x} \in \{0,1\}^{VN}}{\text{minimize}} \quad \sum_{\substack{v \in V \\ u \in V}} \sum_{\substack{i \in N \\ j \in N}} C_{ij,vu} x_{vi} x_{uj} \tag{4.9}$$

$$\text{s.t.} \quad \sum_{n \in N} x_{vn} = 1, \quad x_{vn} = 0, \text{ if } n \notin M_v, \ \forall v \in V, \tag{4.10}$$

where $C_{ij,vu} = \begin{cases} \hat{c}_{ij} d_{vu} & \text{if } i \neq j \\ 0 & \text{otherwise.} \end{cases}$

Consider the condition $b_l > \sum_{e \in E} d_e, \ \forall l \in L$. If this condition is true, then under any embedding the link capacity constraint (4.6) is satisfied. We call this condition "loose capacities".

Lemma 4.1 *MGE and VNE-SF are equivalent problems under the condition of loose capacities.*

The proof of the lemma is straightforward by noticing that under loose capacities, an optimal solution of VNE-SF will always be routed over shortest paths, hence the cost used in the multipartite graph is always the correct one.

4.2.3 The 3-SAT Problem

In this subsection we give a brief reminder of the *NP*–complete [6] 3-SAT problem. Consider N Boolean variables $x_1, \dots, x_N$. A *clause* is a disjunction of literals e.g., $(x_2 \vee \overline{x_4} \vee x_7)$, where a literal is a variable x_i or its logical negation $\overline{x_i}$. In the 3-SAT problem, there is a conjunction of a finite number of clauses, each clause has exactly three literals and the problem is to know if there is any assignment of the variables which results in the expression evaluating to **True**, that is, if the expression is *satisfiable*. E.g., the following expression is satisfiable as can be confirmed by the assignment $x_1 = \textbf{True}, x_2 = \textbf{False}, x_3 = \textbf{True}, x_4 = \textbf{True}$

$$C(\boldsymbol{x}) = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4) \tag{4.11}$$
$$\wedge (\overline{x_2} \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_3 \vee \overline{x_4}), \tag{4.12}$$

which we can write more compactly as

$$C(\boldsymbol{x}) = C_1(\boldsymbol{x}) \wedge C_2(\boldsymbol{x}) \wedge C_3(\boldsymbol{x}) \wedge C_4(\boldsymbol{x})$$

with $C_i(\boldsymbol{x})$ defined in the obvious way.

4.2.4 Complexity of VNE-SF

Theorem 4.2 *The VNE-SF problem is NP–hard.*

Proof: We show that the decision version of the VNE-SF problem is NP-complete by reduction from 3-SAT. We will assume that we are given a black box algorithm that can solve in polynomial time any instance of the VNE-SF, and then we will show that we can apply this black box algorithm on the multipartite graph to decide any instance of the 3-SAT decision problem in polynomial time, which completes the reduction.

Consider an arbitrary instance of the 3-SAT problem $C(\boldsymbol{x}) = \bigwedge_{k=1}^{K} C_k(\boldsymbol{x})$. The multipartite graph $\hat{G} = (\hat{V}, \hat{E}, \hat{\boldsymbol{c}})$ is constructed from the 3-SAT problem instance as follows: Each instance of a literal is given its own variable, meaning that if the literal is in more than one clause, there will be a variable for each of those clauses, and the clauses define the partitions. Thus, with slight abuse of notation, $\hat{V} = \{v_i^k : x_i \in C_k \text{ or } \overline{x_i} \in C_k\}$ (we may assume that no clause contain a literal and its negation since this is always true and can be eliminated from the problem).

The clauses C_k define a partition of the vertex set in the obvious way, and the there is an edge between every pair of nodes that are in different partitions/clauses. The cost of an edge $e = (v_i^k, v_j^l)$ is 0 unless $i = j$ and they represent negations of each other, i.e., v_i^k represents x_i (or $\overline{x_i}$) and v_i^l represents $\overline{x_i}$ (x_i). In the latter case, the cost is 1. Hence, we give cost 1 to edges that connect literals that cannot both be true at the same time otherwise give it cost 0 as shown in the figure.

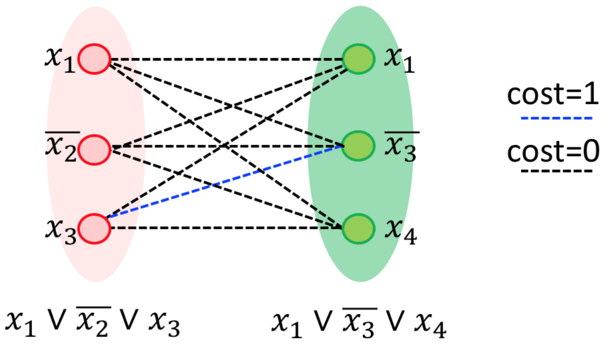
x_1
$\overline{x_2}$
x_3
x_1
$\overline{x_3}$
x_4
cost=1
cost=0
$x_1 \vee \overline{x_2} \vee x_3$
$x_1 \vee \overline{x_3} \vee x_4$

Figure 4.4: Multipartite graph for two clauses of a 3-SAT instance.

Since the number of clauses in the expression $C(\boldsymbol{x})$ is at most polynomial to N, the multipartite graph for all the clauses can be built in polynomial time as well.

Now the question “is there a $\boldsymbol{x} \in \{0,1\}^N$ that satisfies $C(\boldsymbol{x}) = 1$?” can be answered in the following manner. We create the multipartite graph as above, and then call the given black box algorithm, which provides the minimum cost embedding in polynomial time. If the cost of the embedding is 0, the answer is YES, if the cost is positive, the answer is NO.

To verify this, note that a zero cost embedding allows a selection of a node at each clause that can be set to 1 without any conflicts with any other clause. Additionally, if the minimum cost is positive, it means that there exists no zero cost embedding, and hence no conflict-free allocation to make all clauses 1. ■

References

[1] B. Addis, D. Belabed, M. Bouet, and S. Secci. Virtual network functions placement and routing optimization. In *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, pages 171–177, Oct 2015.

[2] B. Addis, D. Belabed, M. Bouet, and S. Secci. Virtual network functions placement and routing optimization. In *CloudNet*, 2015.

[3] D. Bhamare, R. Jain, M. Samaka, and A. Erbad. A survey on service function chaining. *J. Netw. Comput. Appl.*, pages 138–155, 2016.

[4] M. Chowdhury, F. Samuel, and R. Boutaba. Polyvine: policy-based virtual network embedding across multiple domains. In *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, pages 49–56. ACM, 2010.

[5] Cloud Native Computing Foundation (CNCF). Charter document. https://www.cncf.io/about/charter.

[6] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[7] G. Even, M. Medina, and B. Patt-Shamir. On-line path computation and function placement in sdns. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 131–147. Springer, 2016.

[8] G. Even, M. Rost, and S. Schmid. An approximation algorithm for path computation and function placement in sdns. In *International Colloquium on Structural Information and Communication Complexity*, pages 374–390. Springer, 2016.

[9] S. Even, A. Itai, and A. Shamir. On the complexity of time table and multi-commodity flow problems. In *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pages 184–193, Oct 1975.

[10] H. Feng, J. Llorca, A. Tulino, D. Raz, and A. Molisch. Approximation algorithms for the NFV service distribution problem. In *INFOCOM*, 2017.

[11] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach. Virtual network embedding: A survey. *IEEE Communications Surveys Tutorials*, pages 1888–1906, 2013.

[12] M. Gao, B. Addis, M. Bouet, and S. Secci. Optimal orchestration of virtual network functions. *arXiv:1706.04762*, 2017.

[13] A. Garcia-Saavedra, X. Costa-Perez, D. J. Leith, and G. Iosifidis. Fluidran: Optimized vran/mec orchestration.

[14] A. Gupta, M. F. Habib, P. Chowdhury, M. Tornatore, and B. Mukherjee. Joint virtual network function placement and routing of traffic in operator networks. In *NetSoft*, 2015.

[15] J. Kleinberg. *Approximation algorithms for disjoint paths problems*. PhD thesis, Dept. of EECS, MIT, 1996.

[16] Kubernetes Project. Reference documentation. https://kubernetes.io/ docs/reference/.

[17] M. Leconte, G. Paschos, P. Mertikopoulos, and U. Kozat. A resource allocation framework for network slicing. In *INFOCOM*, 2018. submitted.

[18] A. Maeder, M. Lalam, A. D. Domenico, E. Pateromichelakis, D. Wubben, J. Bartelt, R. Fritzsche, and P. Rost. Towards a flexible functional split for cloud-ran networks. *EuCNC*, 2014.

[19] M. Mechtri, C. Ghribi, and D. Zeghlache. A scalable algorithm for the placement of service function chains. *IEEE Transactions on Network and Service Management*, pages 533–546, 2016.

[20] M. R. Rahman and R. Boutaba. SVNE: Survivable virtual network embedding algorithms for network virtualization. *IEEE Transactions on Network and Service Management*, pages 105–118, 2013.

[21] M. Rost and S. Schmid. $\mathcal{NP}$–completeness and inapproximability of the virtual network embedding problem and its variants. Technical report, arXiv:1801.03162, 2018.

[22] S. Vassilaras, L. Gkatzikis, N. Liakopoulos, I. Stiakogiannakis, M. Qi, L. Shi, L. Liu, M. Debbah, and G. Paschos. The algorithmic aspects of network slicing. *IEEE Communication Magazine*, 2017.

[23] S. Vassilaras, L. Gkatzikis, N. Liakopoulos, I. N. Stiakogiannakis, M. Qi, L. Shi, L. Liu, M. Debbah, and G. S. Paschos. The algorithmic aspects of network slicing. *IEEE Communications Magazine*, 55(8):112–119, 2017.

[24] P. Vyavahare, N. Limaye, and D. Manjunath. Optimal embedding of functions for in-network computation: Complexity analysis and algorithms. *IEEE/ACM Transactions on Networking*, 2016.

[25] T. Wang and M. Hamdi. Presto: Towards efficient online virtual network embedding in virtualized cloud data centers. *Computer Networks*, 106:196–208, 2016.

[26] M. Yu, Y. Yi, J. Rexford, and M. Chiang. Rethinking virtual network embedding: substrate support for path splitting and migration. *ACM SIGCOMM Computer Communication Review*, 38(2):17–29, 2008.