Network Slicing with Splittable Flows is Hard

Georgios S. Paschos, Mohammed Amin Abdullah, and Spyridon Vassilaras
Mathematical and Algorithmic Sciences Lab, Paris Research Center, Huawei Technologies, France
{georgios.paschos, mohammed.abdullah, spyros.vassilaras}@huawei.com

***Abstract*—Allocating resources to network slices can be achieved by means of solving virtual network embedding problems, whereby virtual nodes are used to reserve computing resources on cloud nodes, and virtual links are used to reserve bandwidth resources on network paths. Since the associated optimization problem is also NP-hard to approximate, in this paper we focus on a natural simplified setting of interest: the case where the tunnels can be embedded with splittable flows. For this problem, we provide a simple proof that it is NP-hard by a reduction from the 3-SAT problem. Further, using the idea of the *multipartite graph*, we propose a poly-time heuristic for the loose capacity constraint case, based on linear relaxation and randomized rounding. This heuristic is shown to have small optimality gaps in extensive simulations.**

I. Introduction

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 [10]. 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, cf. Fig. 1. This problem is of supreme importance for future edge clouds, but also of combinatorial nature and therefore quite difficult to solve; with node embedding fixed, VNE-UF becomes equivalent to the well-known *minimum cost unsplittable multicommodity flow* which is *NP*–hard [14].

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 *Software Defined Network* (SDN) architectures [24]. Hence, in this paper we 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 [4], [15], but it creates the overhead of maintaining the state consistency between the different parts of the VNF, and hence we leave this consideration for future work. 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. However, the actual complexity of the VNE-SF problem is unknown.

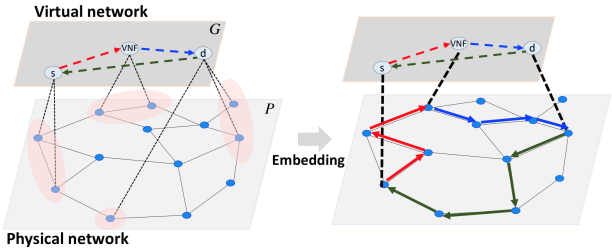

Fig. 1. Virtual network embedding with unsplittable flows.

In this paper, we prove that the VNE-SF remains *NP*–hard by a novel reduction from 3-SAT. Our reduction is based on the idea of relaxing the link capacities and then studying the VNE-SF as an assignment problem on a multipartite graph where the link costs can be computed as shortest paths. Using the same multipartite graph we then propose a heuristic method based on linear relaxation and randomized rounding, which works for VNE-UF and VNE-SF with loose capacity constraints and is shown to perform well in extensive simulations.

The embedding of virtual networks (both splittable and unsplittable variants) is a very important research topic for future edge clouds. Soon network functionality like firewalls, caches, and authentication will be performed by software middleboxes, called *Virtual Network Functions* (VNFs). The VNFs will run at different cloud locations on general purpose computing hardware, and traffic will be steered among them remotely using *Software Defined Networks*. Combining these two enabling technologies, it will be possible to launch applications *when-and-where*, satisfying diverse and demanding requirements, in the form of virtual networks called *network slices* [21]. 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.

In this context, the VNE-UF problem is a fundamental mathematical problem for provisioning resources for network slices. Unfortunately one major bottleneck in the above plan is the complexity of the VNE-UF problem. Our contribution in this paper is thus to further the understanding of the complexity of this problem and to propose efficient solutions. Specifically:

- We formulate the VNE-SF problem with splittable flows, and we show it is *NP*–hard by a reduction from 3SAT.
- Assuming large-enough capacities, we provide a local search algorithm that always converges to a local minimum in polynomial time.

II. System Model

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 flows: 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:

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

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.

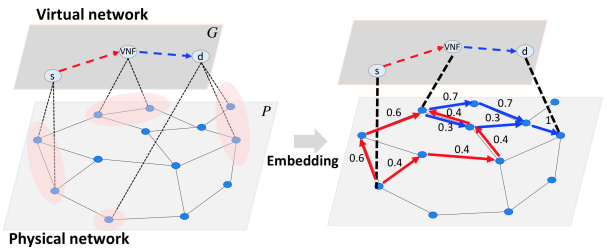

Fig. 2. An example of a virtual network embedding with splittable flows. The vLink demands are split over multiple paths, which constitute a splittable flow.

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 Flows* (VNE-UF), formalized as:

VNE-UF

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

$$\text{subject to} \quad \sum_{e \in E} d_e y_{el} \leq b_l, \ \forall l \in L, \tag{2}$$

$$\sum_{n \in N} x_{vn} = 1, \ \ x_{vn} = 0, \text{ if } n \notin M_v, \ \forall v \in V, \tag{3}$$

$$\sum_j y_{e(n,j)} - \sum_j y_{e(j,n)} = x_{v_1 n} - x_{v_2 n}, \forall n \in N. \tag{4}$$

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 (2) is the link capacity constraint, constraint (3) ensures that vNode v is embedded only once and only at available options M_v, while constraint (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 [21] (ii) the VNF chaining problem [2], (iii) the joing optimization of routing and VNF placement [1], (iv) the optimal functional split in C-RAN [17], (v) the embedding of computation graphs on networks [22], and (v) the minimum cost multicommodity flow problem [8]. In extended variants, it might be useful to consider (i) multiple virtual networks [3], (ii) QoS and survivability constraints [21], as well as (iii) online variants [6], [23].

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 [8]. In parallel to the writing of this paper, [20] appeared, which claims that VNE-UF is APX-hard, i.e., it is *NP*–hard to approximate within a constant factor.

A. VNE for Splittable Flows

In this paper 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, as illustrated in Fig. 2. The corresponding problem becomes:

VNE-SF

$$\underset{\substack{\boldsymbol{x}\in\{0,1\}^{VN}\\ \boldsymbol{y}\in[0,1]^{EL}}}{\text{minimize}} \quad \sum_{\substack{e\in E\\ l\in L}} c_l d_e y_{el} \tag{5}$$

$$\text{subject to} \quad \sum_{e\in E} d_e y_{el} \leq b_l,\ \forall l \in L, \tag{6}$$

$$\sum_{n\in N} x_{vn} = 1,\ \ x_{vn} = 0,\ \text{if } n \notin M_v,\ \forall v \in V, \tag{7}$$

$$\sum_j y_{e(n,j)} - \sum_j y_{e(j,n)} = x_{v_1 n} - x_{v_2 n}, \forall n \in N. \tag{8}$$

B. Related Work

First, we discuss the importance of the VNE-UF problem. When orchestrating optical networks, VNE-UF is used to map an electrical network configuration onto the optical infrastructure [19]. Typically, in this procedure, further constraints should be added, e.g. Quality of Service constraints, resiliency, and optical wavelength allocations. Hence, the VNE-UF serves in this case as a cornerstone problem, and its complexity affects also the corresponding generalizations. In the recent literature of edge computing, a great deal of research is devoted to the problem of mapping Virtual Network Functions arranged in a chain, studied under the name of VNF chaining [2]. This problem is a special case of VNE-UF where the virtual request graph G is a simple line network. In fact, due to the graph restriction this problem is simpler and admits efficient approximations [7]. In future virtualized mobile networks such as C-RAN, the goal is to decouple mobile network processing functionality from transmitting capabilities and economize the networks using commodity hardware-based data centers to host the functionality and dumb remote radio heads to transmit. Prior work studies the optimal functional splitting, i.e., the decision of keeping some parts of the network processing at the edge of the network and pushing the rest to the cloud [12]. In the upcoming 5G wireless networks, mobile edge computing applications will be provided isolated resources by means of network slices, essentially virtual networks co-existing on the global infrastructure.

A common feature in all the above scenarios is the fact that the solution of the VNE-UF problem is a potential computational bottleneck in performing resource allocation operations. For large-scale virtualized infrastructures, resolving VNE-UF is one of the most important networking challenges for the future. Given that VNE-UF is APX-hard [20], the next question is how to address this problem in an efficient manner. Several past works [1], [2], [10], [11], [13], [18], [19] focused on proposing heuristic algorithms that can obtain a solution in polynomial time, but without any guarantees. Another line of work provides approximation algorithms for special graphs [7], while some papers propose to study the linear relaxation of the problem [9], or studying corresponding problems with convex objectives [16]. The latter approach essentially assumes that resources are divisible in the network. In this paper, we focus on the important case where routing is splittable, but function placement is integral. In section III we will prove that the VNE-SF problem is *NP*–hard. Then in section IV we will propose a polynomial time heuristic algorithm based on linear programming relaxation and randomized rounding.

For completeness, we mention that the work of Rost and Schmidt which appeared in the report [20] in parallel with this work also includes a complexity result for our case. However, our proof is more specific and simpler, and hence of potential interest to the community.

III. Hardness of VNE-SF

A. The Multipartite Graph

Fundamental to our approach is the construction of the *multipartite 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 identify the two node subsets M_{v_1}, M_{v_2} and then we denote with $\hat{E}(l)$ 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 $\hat{E} = \cup_{e\in E}\hat{E}(l)$. 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. 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.

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

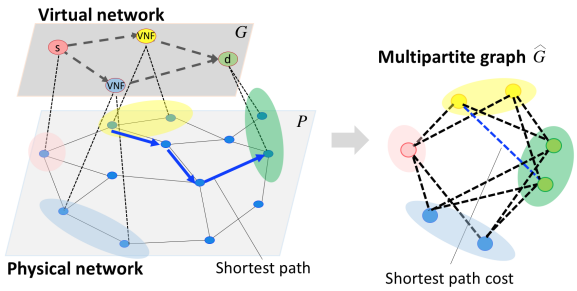

Fig. 3. An example of a multipartite graph construction.

MGE

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

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

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

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 (6) is satisfied. We call this condition "loose capacities".

Lemma 1. *MGE, VNE-SF and VNE-UF 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 (no flow splitting), and hence the cost used in the multipartite graph is always the correct one.

B. The 3-SAT Problem

In this subsection we give a brief reminder of the *NP*–complete [5] 3-SAT problem. Consider N Boolean variables $x_1, \ldots, x_N$. 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 $x_1 = \textbf{True}, x_2 = \textbf{False}, x_3 = \textbf{True}, x_4 = \textbf{True}$

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

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.

C. The Main Result

Theorem 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.

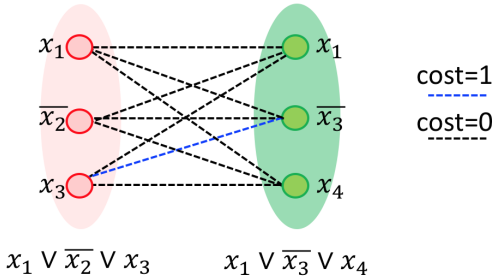

Fig. 4. Multipartite graph for two clauses of a 3-SAT instance.

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. □

IV. Heuristic Polynomial-time Algorithm

In this section, we present a polynomial time heuristic for solving the MGE problem and evaluate the optimality gap between the heuristic and the optimal solution in randomly generated multipartite graphs. We also apply this heuristic to solve the VNE problem with loose capacities for randomly generated VNE problems. Surprisingly the optimality gaps in the second case are much smaller which is an interesting finding, very useful in practical applications. The reason for this phenomenon will be explained later on.

First let us write the MGE problem in the following linear form by introducing the additional variables $y_{ij,vu} = x_{vi}x_{uj}$ which take the value 1 iff node v is embedded on node i and node u on node j:

MGE-IP

$$\underset{\substack{\boldsymbol{x}\in\{0,1\}^{V\hat{V}}\\ \boldsymbol{y}\in\{0,1\}^{\hat{E}}}}{\text{minimize}} \quad \sum_{\substack{v\in V\\ u\in V}}\sum_{\substack{i\in\hat{V}\\ j\in\hat{V}}} C_{ij,vu}y_{ij,vu} \tag{9}$$

$$\text{s.t.} \sum_{n\in\hat{V}} x_{vn} = 1, \;\; x_{vn} = 0, \text{ if } n \notin M_v, \; \forall v \in V, \tag{10}$$

$$\sum_{j\in\hat{V}} y_{nj,vu} - \sum_{j\in\hat{V}} y_{jn,vu} = x_{un} - x_{vn},$$

$$\forall n \in \hat{V}, (u,v) \in \hat{E}, \tag{11}$$

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

Our heuristic starts by solving the LP relaxation of the above problem where both the node embedding x_{ui} and bipartite link selection $y_{ij,vu}$ variables are assumed continuous in $[0, 1]$. The obtained fractional solution is then rounded as follows: among the node embedding variables x_{ui} for the same virtual node u, the largest one is rounded to 1 and the others to 0. In case two or more node embedding variables for the same virtual node are equal (and larger than all the others), the first such variable (i.e., the variable with the lowest i) is rounded to 1 and all others to 0. Note that the values of the $y_{ij,vu}$ variables are uniquely defined by knowing the values of all x_{ui} when considering integer 0,1 solutions.

A local search is then performed to improve the rounded solution. The local search takes all virtual nodes one by one and tries to embed them on different multipartite graph nodes while keeping all other node embeddings fixed. This way we determine the lowest cost embedding for a given node, keep this embedding fixed and proceed to the next virtual node until all nodes have been considered. Obviously, this local search takes polynomial time and since the LP solution and rounding are also of polynomial complexity, the complete heuristic algorithm is polynomial.

A. Numerical results

In order to demonstrate the performance of our heuristic algorithm, we experiment with randomly generated graphs. More specifically, we randomly generate physical and virtual network graph pairs as follows: Physical networks are $G(N, p)$ Erdos-Renyi random graphs, where N is the number of nodes in the graph and p the probability that two nodes are connected. Virtual networks have V vNodes randomly connected so that each vNode has a degree approximately equal to d. Each vLink is assigned a traffic demand d_{uv} drawn uniformly in $[d_{min}, d_{max}]$. Each physical link is assigned a cost c_{ij} drawn uniformly in $[c_{min}, c_{max}]$. Each vNode must be embedded to one of the physical nodes in a constraint set. Constraint sets are disjoint and of the same size S.

For each such VNE problem we construct the associated multipartite graph as described in the previous section, i.e., the cost of each reduced graph link is the cost of the min-cost path between the corresponding nodes in the physical network. We then solve the IP problem (9)-(11) and its LP-relaxation using CPLEX and apply our heuristic algorithm to round the LP-relaxation solution and locally improve it.

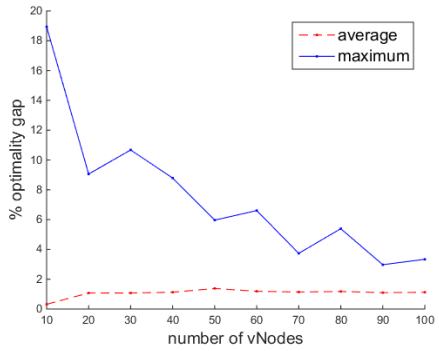

Fig. 5. Average and maximum percent optimality gap between the exact optimal and the solution provided by the heuristic algorithm in case the multipartite link costs are calculated from the min-cost paths in a physical network.

Fig. 5 shows the average and maximum optimality gap between the IP and the heuristic solution over 100 experiments for each value of V. The parameters used for making this plot are as follows: $V \in \{10, 20, \ldots, 100\}$, d = 5, N = 20×V, $p = 0.1$, S = 10, $[d_{min}, d_{max}]$=[2,10], $[c_{min}, c_{max}]$=[4,400]. It can be seen that the achieved optimality gaps are surprisingly low for an NP-hard problem. To discover the reason for this, we repeated the above experiment with the substantial difference that the bipartite graph costs are not calculated as min-path costs on a physical network but are drawn independently from a uniform distribution in $[C_{min}, C_{max}]$.

Fig. 6 shows the average and maximum optimality gap between the IP and the heuristic solution over 100 experiments for each value of $V \in \{10, 20, \ldots, 80\}$, in this case. The parameters for this experiment are the same as

before except that $[C_{min}, C_{max}]$=[4,400]. Obviously the optimality gaps are much larger than before. This shows that the reason for getting such low optimality gaps in the previous experiment is that the resulting costs C when generated from the physical network graph follow a much narrower distribution which results in easier IP problems. Indeed, the histogram of costs C for a single experiment with V=80 is shown in Fig. 7.

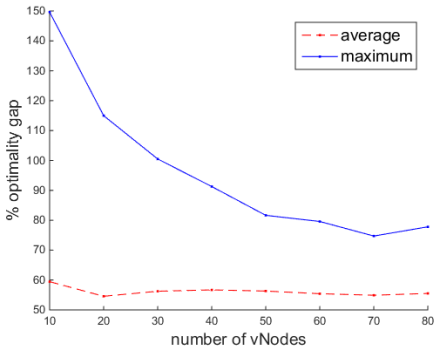

Fig. 6. Average and maximum percent optimality gap between the exact optimal and the solution provided by the heuristic algorithm when the multipartite link costs are independently drawn from a uniform distribution.

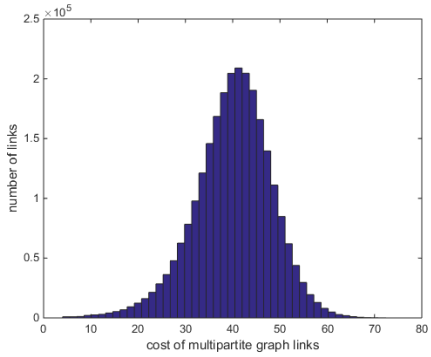

Fig. 7. Histogram of multipartite graph link costs calculated as the min-cost paths in a physical network for a random experiment (V=80).

V. Conclusion

We have shown that the variant VNE-SF of the virtual network embedding problem is NP-hard using a reduction from the 3-SAT problem. This shows that the original VNE-UF problem subsumes two different combinatorial structures, one due to integral routing and one due to integral placement of functions. Fortunately, in VNE-UF and VNE-SF where capacity constraints are relaxed, our heuristic algorithm seems to provide very good performance. Note that this algorithm can be used as a basis to solve VNE problems with tight capacity constraints using Lagrangian relaxation methods.

References

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

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

[3] M. Chowdhury, F. Samuel, and R. Boutaba. Polyvine: policy-based virtual network embedding across multiple domains. In *2nd ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, pages 49–56. ACM, 2010.

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

[5] S. A. Cook. The complexity of theorem-proving procedures. In *3rd annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] M. Gao, B. Addis, M. Bouet, and S. Secci. Optimal orchestration of virtual network functions. *Computer Networks*, 142:108 – 127, 2018.

[12] A. Garcia-Saavedra, X. Costa-Perez, D. J. Leith, and G. Iosifidis. FluidRAN : Optimized vRAN / MEC orchestration. In *IEEE INFOCOM*, 2018.

[13] 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.

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

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

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

[17] 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.

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

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

[20] 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.

[21] 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 Comm. Mag.*, 2017.

[22] 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.

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

[24] 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.