On the Hardness and Inapproximability of Virtual Network Embeddings

Matthias Rost and Stefan Schmid

***Abstract*—Many resource allocation problems in the cloud can be described as a basic Virtual Network Embedding Problem (VNEP): the problem of finding a mapping of a *request graph* (describing a workload) onto a *substrate graph* (describing the physical infrastructure). Applications range from mapping testbeds, over the embedding of batch-processing tasks to the embedding of service function chains and come with different mapping restrictions for nodes and edges. The restrictions studied most often are node and edge capacities, node mapping, edge routing and latency restrictions. While the VNEP has been studied intensively, complexity results are only known for specific models and this paper provides a first comprehensive study of the computational complexity of the VNEP by systematically analyzing its hardness for any combination of the above stated mapping restrictions. For all studied variants the $\mathcal{NP}$-completeness of the respective decision problems is shown. Furthermore, $\mathcal{NP}$-completeness results for finding approximate embeddings, which may, e.g., violate capacity constraints by certain factors, are derived. Lastly, it is also shown that all these results pertain when restricting the request graphs to planar and degree-bounded graphs. While theoretic in nature, our results have severe practical implications. Firstly, *any* optimization variant of the VNEP is $\mathcal{NP}$-hard and cannot be approximated for any of the studied restrictions, unless $\mathcal{P} = \mathcal{NP}$. Secondly, we uncover structural hardness properties: the VNEP is $\mathcal{NP}$-hard and inapproximable even if, e.g., only node placement and edge routing restrictions are considered.**

***Index Terms*—Network virtualization, virtual network embeddings, computational complexity, inapproximability.**

I. Introduction

At the heart of the cloud computing paradigm lies the idea of efficient resource sharing: due to virtualization, multiple workloads can co-habit and use a given resource infrastructure simultaneously. Indeed, cloud computing introduces great flexibilities in terms of *where* workloads can be mapped. At the same time, exploiting this mapping flexibility poses a fundamental algorithmic challenge. In particular, in order to provide predictable performance, guarantees on all, i.e., node and edge, resources need to be ensured as cloud application performance can otherwise suffer significantly [2].

The underlying algorithmic problem is essentially a graph theoretical one: both the workload as well as the infrastructure can be modeled as *graphs*. The former, the so-called *request graph*, describes the resource requirements both on the nodes (e.g., the virtual machines) as well as on the interconnecting network. The latter, the so-called *substrate graph*, describes the physical infrastructure and its resources (servers and links). The task is to embed the request graph on the substrate network by mapping each request node on a substrate node and embedding each request edge via a path in the substrate network connecting the respective endpoints while satisfying capacity constraints (see Figure 1 for an example).

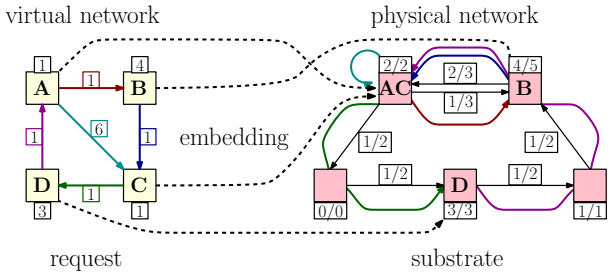

Fig. 1. Example of embedding a virtual network request (left) on the physical substrate network (right). The numeric labels of the request graph's network elements denote the resource demands, while for the substrate network the resource usage and the total capacity are given. In the embedding, each request node is mapped to a single substrate node and each request edge is realized by a path in the substrate network. Notably, the model allows for collocation of virtual nodes **A** and **C** on the same substrate node and, accordingly, the request edge $(\mathbf{A}, \mathbf{C})$ does not use any of the substrate edges.

The problem is known in the networking community under the name *Virtual Network Embedding Problem* (VNEP). It has been studied intensively for over a decade in various contexts and subject to varying objectives and several additional constraints [3]. The objectives most prominently studied in the online setting are to minimize resource allocations or to balance the load while in the offline setting the task is mainly to maximize the profit. While node and edge capacities are nearly always enforced, node placement, routing and latency restrictions are often considered as well [3]. Besides the VNEP, the same graph theoretic problem is considered in the context of Service Function Chaining [4], [5] and the embedding of virtual clusters [6], [7]. Specifically, service chains have been primarily envisioned in the context of stitching virtualized network functions as, e.g., firewalls in data center or ISP networks, while virtual clusters were envisioned in the context of batch-processing applications. As the name suggests, service chains model chain-like request abstractions while virtual clusters constrain request topologies to simple star-shaped networks.

This work presents fundamental computational complexity results for the VNEP and its related problems. In contrast to

Manuscript received May XX, XXXX; revised February XX, XXXX; accepted February XX, XXXX; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor X. XXXXXXX. Date of publication June XX, XXXX; date of current version August XX, XXXX. This work and its dissemination efforts were supported in part by the BMBF Software Campus grant 01IS1205 and the European Research Council (ERC) grants ResolutioNet (ERC-StG-679158) and AdjustNet (ERC-COG-864228). Parts of this work were presented at IFIP Networking 2018 [1].
Matthias Rost is with the Faculty of Electrical Engineering and Computer Science of the Technische Universität Berlin, 10587 Berlin, Germany.
Stefan Schmid is with the Faculty of Computer Science of the University of Vienna, 1090 Vienna, Austria.

previous works, this work mainly studies the decision variant of the VNEP, where the task is to *decide* whether an embedding exists that meets all imposed constraints. Notably, any result for the decision VNEP carries over to any optimization variant of the VNEP under the same restrictions. Specifically, optimizing over the set of feasible solutions necessitates solving the respective decision problem by constructing a feasible solution. To clearly discern between the decision and optimization variants of the VNEP, we refer to these as D-VNEP and O-VNEP. Before discussing our results, we review the related work in the following.

A. Related Work

a) Objectives & Restrictions: Depending on the setting, many different objectives are considered for the VNEP. The most studied ones concern minimizing the (resource allocation) cost [3], [8], maximizing the profit by exerting admission control [9], [10], and minimizing the maximal load [4], [8].

Besides commonly enforcing that the substrate's physical node and edge capacities are not exceeded to provide Quality-of-Service [3], additional restrictions have emerged:

- Restrictions on the placement of virtual nodes constrain the potential mapping of each virtual node to a subset of substrate nodes. This type of restriction first arose to enforce closeness to locations of interest [8], but were also used in the context of privacy policies, e.g., to restrict mappings to certain countries [11]. Additionally, these restrictions are now also used in the context of Service Function Chaining, as, e.g., specific functions may only be mapped on commodity servers, while, e.g., virtualized firewall appliances may not [4], [5].
- Routing restrictions constrain the set of substrate edges which may be used to map virtual edges. These first arose in the context of expressing security policies, as for example some traffic may not be routed via insecure domains or physical links shall not be shared with other virtual networks [3], [12].
- Restrictions on latencies, i.e., imposing bounds on the latency of edge embeddings, were studied for the VNEP in [13] and have been recently studied intensely in the context of Service Function Chaining to achieve responsiveness and Quality-of-Service [4], [5].

b) Algorithmic Approaches: Dozens of algorithms were proposed to solve the VNEP [3] and its siblings, including the embedding of virtual clusters [6] and of service function chains [4]. Most approaches to solve the VNEP rely on (meta-)heuristics [3]. On the other hand, several works study exact (non-polynomial time), algorithms to compute (near-)optimal solutions. Here, Mixed-Integer Programming is the most widely used approach [4], [10], [13].

Approximation algorithms providing quality guarantees for the VNEP have only been recently presented for the offline setting in which the substrate provider can select which requests to embed to maximize its profit. In particular, the embedding of service chains is approximated under assumptions on the requested resources and the achievable benefit in [14]. In [15] the first approximations for cyclic request graph topologies, namely cactus request graphs, was detailed under node placement and routing restrictions. In [16] these results were extended to *arbitrary* request graph topologies while also taking latency restrictions into account. Importantly, the runtime of the approximation for general request graphs depends exponentially on the complexity of the request graphs, measured via their treewidth. Our results validate that this is indeed the best one can hope for, as it is shown that the VNEP even remains inapproximable in polynomial-time for planar request graphs.

c) Complexity Results: Surprisingly, despite the relevance of the problem and the large body of literature, the complexity of Virtual Network Embedding Problem has not received much attention. While it is not hard to see that the VNEP generalizes several $\mathcal{NP}$-hard problems as, e.g., the k-disjoint paths problem [17], the minimum linear arrangement problem [18], or the subgraph isomorphism problem [19], most works on the VNEP cite a $\mathcal{NP}$-hardness result contained in a technical report from 2002 by Andersen [20].

The only other work studying the computational complexity is one by Amaldi et al. [21], which proved the $\mathcal{NP}$-hardness and inapproximability of the profit maximization objective while not taking into account latency or routing restrictions and not considering the hardness of embedding a *single* request. In contrast, this work provides detailed results for any combination of the above introduced node and edge mapping restrictions. Furthermore, our reduction framework allows to show the inapproximability of approximate embeddings, i.e., when relaxing constraints, and under restrictions of the request graphs to planar graphs.

Lastly, we note that the VNEP is not only related to the aforementioned classical optimization problems, but that similar tasks arise also in other networking contexts as embedding coflows [22] or routing packets through virtualized pipelines [23]. These problems exhibit similar properties as the VNEP but generally do not reduce to the VNEP. Routing through pipelines is an inherent temporal task, while VNEP embeddings remain the same over the embedding period and for coflows the workload endpoints are assumed to be fixed.

B. Contributions and Overview

This work initiates the systematic study of the computational complexity of the VNEP. Taking all the aforementioned restrictions into account, first a concise taxonomy of the VNEP variants is compiled in Section II. Then, a powerful reduction framework from 3-SAT is presented in Section III, which is the base for most hardness results presented in this paper. In particular, we show the following (see also Table I):

- We show the $\mathcal{NP}$-completeness of the D-VNEP under any combination of the studied node and edge mapping restrictions. Specifically, we study node capacities and node placement restrictions as well as edge capacities, routing, and latency restrictions. Accordingly, for all six different restriction combinations, the $\mathcal{NP}$-completeness of D-VNEP is shown in Section IV.
- We extend these results in Section V and show that the considered variants remain hard even when computing

TABLE I
OVERVIEW ON RESULTS

VNEP variants	Identifier according to Definition 9	⟨**VE**|-⟩	⟨**E**|**N**⟩	⟨**V**|**R**⟩	⟨-|**NR**⟩	⟨-|**NL**⟩	⟨**V**|**L**⟩
	Enforcing node capacities	✓	⋆	✓	⋆	⋆	✓
	Enforcing edge capacities	✓	✓	⋆	⋆	⋆	⋆
	Enforcing node placement restrictions	⋆	✓	⋆	✓	✓	⋆
	Enforcing edge routing restrictions	⋆	⋆	✓	✓	⋆	⋆
	Enforcing latency restrictions	⋆	⋆	⋆	⋆	✓	✓

Results								
Section IV	$\mathcal{NP}$-Completeness of D-VNEP	Thm. 22	Thm. 23	Thm. 24	Thm. 25	Thm. 25	Thm. 26
Section V	$\mathcal{NP}$-Completeness of D-VNEP under node capacity violations by factor $\alpha < 2$	Thm. 27	-	Thm. 27	-	-	Thm. 27
	Hardness of D-VNEP under edge capacity violations by factor $\beta \in \Theta(\frac{\log \lvert V_S \rvert}{\log \log \lvert V_S \rvert})$	Thm. 33	Thm. 31	-	-	-	-
	$\mathcal{NP}$-Completeness of D-VNEP under latency bound violations by factor $\gamma < 2$	-	-	-	-	Thm. 28	Thm. 28
Section VI	All above $\mathcal{NP}$-completeness results are preserved for acyclic substrates	Obs. 34					
	All above results are preserved for acyclic, planar, degree-bounded requests	Thm. 37					
Section VII	All above results translate to the ($\mathcal{NP}$-)hardness and inapproximability of O-VNEP under the respective restrictions and *under any objective*	Thm. 38					

approximate embeddings, which may exceed latency or capacity constraints by certain factors.

- In Section VI it is then shown that the respective D-VNEP variants remain $\mathcal{NP}$-complete even when restricting substrate graphs to acyclic graphs and request graphs to acyclic planar, degree-bounded graphs.
- Given the above results for the decision variants, the $\mathcal{NP}$-hardness and inapproximability of any O-VNEP optimization variant is discussed in Section VII.

Table I summarizes our results and is to be read as follows. Any of the six rightmost columns represents a specific VNEP variant. The ✓ symbol indicates enforced restrictions, while the ⋆ symbol indicates that a restriction is not enforced. Importantly, enabling a ⋆ restriction, does not change the results (cf. Lemma 10 in Section II). Considering a specific variant, the respective column should be read from top to bottom. For example, for ⟨**VE**|-⟩, its $\mathcal{NP}$-completeness is shown in Theorem 22, while results pertaining to computing approximate embeddings are stated in Theorems 27 and 33. Whenever a result is not applicable for a VNEP variant, this is marked using '-'. Notably, the results stated in the last three rows hold for any VNEP variant.

II. FORMAL MODEL

In this section the variants of the VNEP are formalized. We first introduce the capacitated VNEP and then formalize other restrictions, yielding our taxonomy of the VNEP. Furthermore, the notion of approximate embeddings and an Integer Program to solve the D-VNEP are given.

Notation: The following notation is used throughout this work. We use $[x]$ to denote the set $\{1, 2, \ldots, x\}$ for $x \in \mathbb{N}$. For a directed graph $G = (V, E)$, we denote by $\delta^+(v) \subseteq E$ and $\delta^-(v) \subseteq E$ the outgoing and incoming edges of $v \in V$. When considering functions on tuples, we omit the parentheses of the tuple and simply write $f(a, b)$ instead of $f((a, b))$.

A. Basic Problem Definition

We refer to the physical network as substrate network and model it as directed graph $G_S = (V_S, E_S)$. Capacities in the substrate are given by the function $c_S : V_S \cup E_S \to \mathbb{R}_{\geq 0} \cup \{\infty\}$. The capacity $c_S(u)$ of node $u \in V_S$ may represent for example the number of CPU cores while the capacity $c_S(u, v)$ of edge $(u, v) \in E_S$ represents its available bandwidth. By allowing to set substrate capacities to ∞, the capacity constraints of the respective substrate elements can effectively be disabled. We denote by $\mathcal{P}_S$ the set of all simple paths in G_S.

A request is analogously modeled as directed graph $G_r = (V_r, E_r)$ together with node and edge capacities (demands) $c_r : V_r \cup E_r \to \mathbb{R}_{\geq 0}$.

The general task is to find a *mapping* of the request graph G_r to the substrate network G_S, i.e., to map each request node to a substrate node and to map each request edge to paths in the substrate. Virtual nodes and edges can only be mapped on substrate nodes and edges of sufficient capacity. Accordingly, we denote by $V_S^i = \{u \in V_S | c_S(u) \geq c_r(i)\}$ the set of substrate nodes supporting the mapping of node $i \in V_r$ and by $E_S^{i,j} = \{(u, v) \in E_S | c_S(u, v) \geq c_r(i, j)\}$ the substrate edges supporting the mapping of virtual edge $(i, j) \in E_r$.

Definition 1 (Valid Mapping)**.** A *valid* mapping of request G_r to the substrate G_S is a tuple $m = (m_V, m_E)$ of functions that map nodes and edges, respectively, s.t. the following holds:

- The function $m_V : V_r \to V_S$ maps virtual nodes to *suitable* substrate nodes, such that $m_V(i) \in V_S^i$ holds for $i \in V_r$.
- The function $m_E : E_r \to \mathcal{P}_S$ maps virtual edges $(i, j) \in E_r$ to paths in G_S connecting $m_V(i)$ to $m_V(j)$, such that $m_E(i, j) \subseteq E_S^{i,j}$ holds for $(i, j) \in E_r$. □

Accordingly, a valid mapping ensures that virtual nodes and virtual edges are only mapped to nodes of sufficient capacity and that virtual edges correctly connect the respective endpoints. Furthermore, note the following. Firstly, the mapping $m_E(i, j)$ of the virtual edge $(i, j) \in E_r$ may be the empty path, if and only if both i and j are mapped on the same substrate node. Secondly, the definition only enforces that each single resource mapping does not exceed the available capacity. To enforce that the cumulative allocations respect capacities, we formalize the following notion of allocations.

Definition 2 (Allocations)**.** We denote by $A_m(x) \in \mathbb{R}_{\geq 0}$ the resource allocations induced by the valid mapping

$m = (m_V, m_E)$ on substrate element $x \in G_S$, where

$$A_m(u) = \sum_{i \in V_r : m_V(i) = u} c_r(i)$$
$$A_m(u,v) = \sum_{(i,j) \in E_r : (u,v) \in m_E(i,j)} c_r(i,j)$$

for node $u \in V_S$ and edge $(u,v) \in E_S$, respectively. □

We call a mapping *feasible*, if the (cumulative) allocations do not exceed the capacity of any substrate element.

Definition 3 (Feasible Embedding)**.** A mapping m is a feasible embedding, if the allocations do not exceed the capacity, i.e., $A_m(x) \leq c_S(x)$ holds for $x \in G_S$. □

While the related work considers the optimization of embeddings subject to objectives, in this paper mostly the *decision* variant of the VNEP, referred to as D-VNEP, is studied. The following definition caters to the capacitated VNEP, while additional restrictions can be easily included by adapting the notion of valid mappings and feasible embeddings (see below).

Definition 4 (D-VNEP)**.** Given is a single request graph G_r that shall be embedded on the substrate graph G_S. The task is to find any feasible embedding or to decide that no feasible embedding exists. □

Given the decision variant, we also introduce the optimization variant O-VNEP under an abstract objective as follows.

Definition 5 (O-VNEP)**.** Given is a single request G_r that shall be embedded on the substrate G_S and an objective function. The task is to find a feasible embedding of optimal objective or to decide that no feasible embedding exists. □

Note that the optimization variant O-VNEP is introduced with respect to a single request graph, but may also be easily extended to several request graphs.

B. Variants of the VNEP *& Nomenclature*

As discussed when reviewing the related work in Section I-A, additional restrictions are enforced in many settings. Accordingly, we now formalize (i) node placement, (ii) edge routing, and (iii) latency restrictions. Node placement and edge routing restrictions effectively exclude potential mapping options for nodes and edges. For latency restrictions we introduce latency bounds for each of the virtual edges.

Definition 6 (Node Placement Restrictions)**.** For each virtual node $i \in V_r$ a set of forbidden substrate nodes $\overline{V}_S^i \subset V_S$ is provided. Accordingly, the set of allowed nodes V_S^i is set to be $\{u \in V_S \setminus \overline{V}_S^i \mid c_S(u) \geq c_r(i)\}$ for $i \in V_r$. □

Definition 7 (Routing Restrictions)**.** For each virtual edge $(i,j) \in E_r$ a set of forbidden substrate edges $\overline{E}_S^{i,j} \subseteq E_S$ is given. Accordingly, the set of allowed edges $E_S^{i,j}$ is set to be $\{e \in E_S \setminus \overline{E}_S^{i,j} \mid c_S(e) \geq c_r(i,j)\}$ for $(i,j) \in E_r$. □

Definition 8 (Latency Restrictions)**.** For each substrate edge $e \in E_S$ the edge's latency is given via $l_S(e) \in \mathbb{R}_{\geq 0}$. Latency bounds for virtual edges are specified via the function $l_r : E_r \to \mathbb{R}_{\geq 0} \cup \{\infty\}$, such that the latency along the substrate path $m_E(i,j)$, used to realize the edge $(i,j) \in E_r$, must be less than $l_r(i,j)$. Formally, the definition of feasible embeddings (cf. Definition 3) is extended by including that $\sum_{e \in m_E(i,j)} l_S(e) \leq l_r(i,j)$ holds for $(i,j) \in E_r$. □

The following taxonomy is employed to denote the various VNEP variants under the respective restrictions.

Definition 9 (Taxonomy)**.** We use the notation $\langle \mathbf{C} \mid \mathbf{A} \rangle$ to indicate whether and which of the capacity constraints **C** and which of the additional constraints **A** are enforced.

C We denote by **V** node capacities, by **E** edge capacities, and by **-** that none are enforced. When node or edge capacities are *not* considered, we set the capacities of the respective substrate element sets to ∞.

A For additional restrictions **-**, **N**, **L**, and **R** stand for none, node placement, latency, and routing restrictions. □

Hence, $\langle$ **VE** | **-** $\rangle$ refers to the capacitated VNEP without additional constraints and $\langle$ **-** | **NL** $\rangle$ indicates the combination of node placement and latency restrictions without considering substrate capacities. We note that the introduction of more restrictions only makes the respective problem harder:

Lemma 10. *A VNEP variant* $\langle \mathbf{A} \mid \mathbf{C} \rangle$ *that encompasses all restrictions of* $\langle$ **A'** | **C'** $\rangle$ *is at least as hard as* $\langle$ **A'** | **C'** $\rangle$.

Proof. All restrictions were formulated in such a way that these can be easily disabled. Considering capacities and latencies, one may set the respective substrate capacities to ∞ and the latencies of substrate edges to 0, respectively, to disregard these. For node placement and edge restrictions one may set the forbidden node and edge sets to the empty set. Hence, there exists a trivial reduction from $\langle \mathbf{A} \mid \mathbf{C} \rangle$ to $\langle$ **A'** | **C'** $\rangle$. ■

C. Relaxing Constraints: Approximate Embeddings

Within this work, we show the D-VNEP to be $\mathcal{NP}$-complete under any combination of studied node and edge restrictions. As shown in Section VII, this in turn also implies the inapproximability of the respective O-VNEP variants unless $\mathcal{P} = \mathcal{NP}$. Hence, it is natural to consider a broader class of (approximation) algorithms that may violate constraints by a certain factor: instead of answering the question whether an embedding exists that satisfies the capacity constraints, one might relax the capacity constraints and seek embeddings of bounded capacity violations. We refer to these as approximate embeddings:

Definition 11 (α- / β- / γ-Approximate Embeddings)**.** A mapping m is an approximate embedding, if it is valid and violates capacity or latency constraints only within a certain bound. Specifically, we call an embedding α- and β-approximate, when node and edge allocations are bounded by α and β times the respective node or edge capacity. Considering latency restrictions, we call a mapping γ-approximate when latencies are within a factor of γ of the original bound. Formally, the following must hold for $\alpha, \beta, \gamma \geq 1$:

$$\begin{aligned} A_m(u) &\leq \alpha \cdot c_S(u) && \forall u \in V_S \\ A_m(u,v) &\leq \beta \cdot c_S(u,v) && \forall (u,v) \in E_S \\ \sum_{e \in m_E(i,j)} l_S(e) &\leq \gamma \cdot l_r(i,j) && \forall (i,j) \in E_r \end{aligned}$$ □

Given the above notion of approximate embeddings, the approximate D-VNEP and O-VNEP are defined as follows.

Definition 12 (Approximate D-VNEP)**.** Given is a single request graph G_r that shall be embedded on the substrate graph G_S. The task is to return an (α-/β-/γ-)approximate embedding, if a feasible embedding subject to the original restrictions exists, or to decide that none exists. □

Definition 13 (Approximate O-VNEP)**.** Given is a single request graph G_r that shall be embedded on the substrate graph G_S. The task is to return an (α-/β-/γ-)approximate embedding of optimal objective, if a feasible embedding subject to the original restrictions exists, or to decide that none exists. □

D. Integer Programming Formulation

We now give an Integer Programming (IP) formulation, which can be used to solve any of the considered decision VNEP variants. A similar formulation was proposed in [13]. Given the hardness results presented in this paper and given that solving IPs lies in $\mathcal{NP}$ [24], the IP may serve as an attractive approach to solve the respective variants in *exponential* time. Besides the practical application, the existence of our formulation (constructively) shows that the D-VNEP variants considered here are also all contained in $\mathcal{NP}$.

Our formulation naturally encompasses node placement and routing restrictions, while for latencies an additional constraint is introduced. The decision variable $x \in \{0,1\}$ is used to indicate, whether the request graph G_r is embedded or not. By maximizing x, the IP decides whether a feasible embedding exists ($x = 1$) or whether no such embedding exists ($x = 0$). The mapping of virtual nodes is modeled using decision variables $y_i^u \in \{0,1\}$ for $i \in V_r$ and $u \in V_S$. If $y_i^u = 1$ holds, then the virtual node $i \in V_r$ is mapped on substrate node $u \in V_S$. Constraint 2 enforces that each virtual node is mapped to one substrate node, *if* the request is embedded ($x = 1$), while Constraint 3 excludes unsuitable substrate nodes.

For computing edge mappings the decision variables $z_{i,j}^{u,v} \in \{0,1\}$ for $(i,j) \in E_r$ and $(u,v) \in E_S$ are employed.

Integer Program 1: VNEP Decision Variant

$$\max\ x \tag{1}$$

$$\sum_{u \in V_S} y_i^u = x \qquad \forall i \in V_r \tag{2}$$

$$\sum_{u \in V_S \setminus V_S^i} y_i^u = 0 \qquad \forall i \in V_r \tag{3}$$

$$\sum_{(u,v)\in\delta^+(u)} z_{i,j}^{u,v} - \sum_{(v,u)\in\delta^-(u)} z_{i,j}^{v,u} = y_i^u - y_j^u \qquad \forall (i,j) \in E_r, u \in E_S \tag{4}$$

$$\sum_{(u,v)\in E_S \setminus E_S^{i,j}} z_{i,j}^{u,v} = 0 \qquad \forall (i,j) \in E_r \tag{5}$$

$$\sum_{i \in V_r} c_r(i) \cdot y_i^u \le c_S(u) \qquad \forall u \in V_S \tag{6}$$

$$\sum_{(i,j)\in E_r} c_r(i,j) \cdot z_{i,j}^{u,v} \le c_S(u,v) \qquad \forall (u,v) \in E_S \tag{7}$$

$$\sum_{(u,v)\in E_S} l_S(u,v) \cdot z_{i,j}^{u,v} \le l_r(i,j) \qquad \forall (i,j) \in E_r \quad {}^\star \tag{8}$$

If $z_{i,j}^{u,v} = 1$ holds, then the substrate edge (u,v) lies on the path $m_E(i,j)$. Constraints 4 and 5 embed virtual links as paths in the substrate, if the request is embedded. In particular, Constraint 4 constructs a unit flow for virtual edge $(i,j) \in E_r$ from the location $u \in V_S$ onto which i was mapped ($y_i^u = 1$) to the location $v \in V_S$ onto which j was mapped ($y_j^v = 1$), while Constraint 5 excludes unsuitable edges. Constraints 6 and 7 enforce that substrate capacities are obeyed. Lastly, Constraint 8 is only used when latencies are considered: it enforces that the sum of latencies along the embedding path of a virtual edge is smaller than the respective latency bound.

III. Reduction Framework

This section presents the main insight and contribution of this work, namely a generic reduction framework that allows to derive hardness results by slightly tailoring the proof to the respective problem variants. Our reduction framework relies on 3-SAT and we first introduce some notation. Afterwards, a generic construction of D-VNEP instances for 3-SAT formulas is given and the relationship between the existence of specific feasible embeddings and the satisfiability of the 3-SAT formula is established.

*A. 3-*SAT*: Notation and Problem Statement*

We denote by $\mathcal{L}_\phi = \{x_k\}_{k\in[N]}$ a set of $N \in \mathbb{N}$ literals and by $\mathcal{C}_\phi = \{\mathcal{C}_i\}_{i\in[M]}$ a set of $M \in \mathbb{N}$ clauses, in which literals may occur either positively or negated. The formula $\phi = \bigwedge_{\mathcal{C}_i\in\mathcal{C}_\phi} \mathcal{C}_i$ is a 3-SAT formula, iff. each clause $\mathcal{C}_i$ is the disjunction of at most 3 literals of $\mathcal{L}_\phi$. Denoting the truth values by F and T, 3-SAT asks to determine whether an assignment $\alpha : \mathcal{L}_\phi \to \{\mathrm{F},\mathrm{T}\}$ exists, such that ϕ is satisfied. 3-SAT is one of Karp's 21 $\mathcal{NP}$-complete problems:

Theorem 14 (Karp [25])**.** *Deciding 3-*SAT *is $\mathcal{NP}$-complete.*

For reducing 3-SAT to D-VNEP, we assume that the clauses are ordered and we define the following:

Definition 15 (First Occurence of Literals)**.** We denote by $\mathcal{C} : \mathcal{L}_\phi \to [M]$ the function yielding the index of the clause in which a literal first occurs. Hence, if $\mathcal{C}(x_k) = i$, then x_k is contained in $\mathcal{C}_i$ while not contained in $\mathcal{C}_{i'}$ for $i' \in [i-1]$. □

The assignments satisfying *a clause* are defined as follows.

Definition 16 (Satisfying Assignments)**.** We denote by $\mathcal{A}_i = \{a_{i,m} : \mathcal{L}_i \to \{\mathrm{F},\mathrm{T}\} \mid a_{i,m} \text{ satisfies } \mathcal{C}_i\}$ the set of all possible assignments of truth values to the literals $\mathcal{L}_i$ of $\mathcal{C}_i$ satisfying $\mathcal{C}_i$. Note that all elements of $\mathcal{A}_i$ are functions. □

Lastly, to abbreviate notation, we employ $\mathcal{L}_{i,j} = \mathcal{L}_i \cap \mathcal{L}_j$ to denote the intersection of the literal sets of $\mathcal{C}_i$ and $\mathcal{C}_j$.

B. General VNEP *Instance Construction*

For a given 3-SAT formula ϕ, we now construct a D-VNEP instance consisting of a substrate graph $G_{S(\phi)}$ and a request graph $G_{r(\phi)}$. The question whether the formula ϕ is satisfiable will eventually reduce to the question whether a feasible

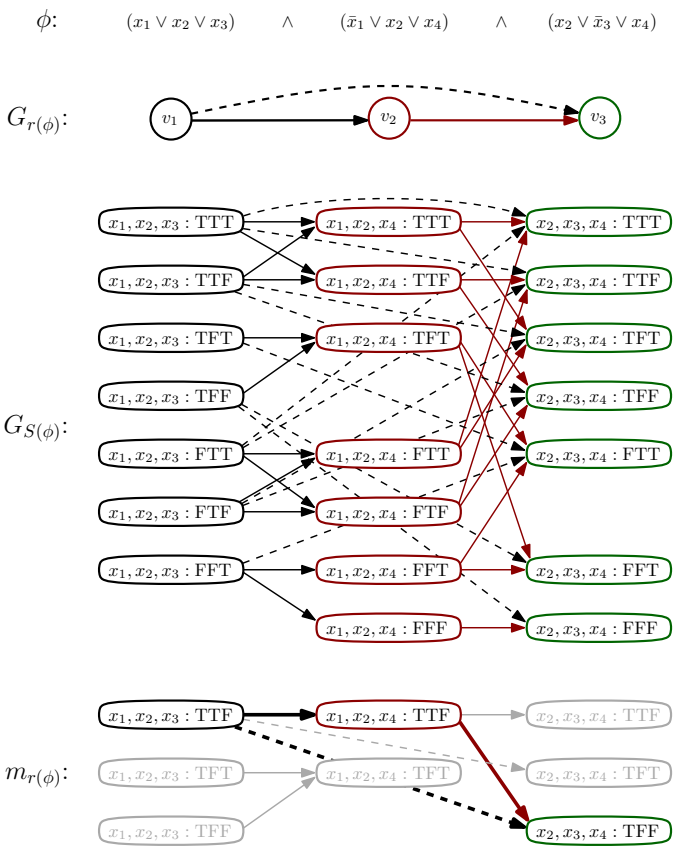

Fig. 2. Visualization of the construction the request and the substrate graphs for the 3-SAT formula ϕ (cf. Definitions 17 and 18). Note that all virtual edges are contained in $E^{\mathcal{L}}_{r(\phi)}$, i.e., these are introduced as the clauses of the connected virtual nodes share literals. Hence, $E^{\mathcal{N}}_{r(\phi)} = \emptyset$ holds and the substrate also does not contain edges for neighboring clauses $E^{\mathcal{N}}_{S}$. Additionally, a mapping $m_{r(\phi)}$ satisfying the conditions of Lemma 20 is shown. Hence, the formula ϕ is satisfiable. Concretely, the mapping corresponds satisfying assignment $x_1 = \mathrm{T}$, $x_2 = \mathrm{T}$, $x_3 = \mathrm{F}$, $x_4 = \mathrm{F}$.

embedding of $G_{r(\phi)}$ on $G_{S(\phi)}$ exists. Figure 2 illustrates the construction described in the following.

Definition 17 (Request Graph $G_{r(\phi)}$)**.** For a given 3-SAT formula ϕ we define the request graph $G_{r(\phi)} = (V_{r(\phi)}, E_{r(\phi)})$ as follows. For each clause $\mathcal{C}_i \in \mathcal{C}_\phi$ a node v_i is introduced, i.e., $V_{r(\phi)} = \{v_i \,|\, \mathcal{C}_i \in \mathcal{C}_\phi\}$. An edge (v_i, v_j) is introduced if either the i-th clause $\mathcal{C}_i$ introduces a literal used in the j-th clause $\mathcal{C}_j$, or if $j = i + 1$ holds. Accordingly, we set $E_{r(\phi)} = E^{\mathcal{L}}_{r(\phi)} \sqcup E^{N}_{r(\phi)}$ with:

$$E^{\mathcal{L}}_{r(\phi)} = \{(v_i, v_j) \,|\, \exists x_k \in \mathcal{L}_{i,j} : \ \mathcal{C}(x_k) = i\}$$
$$E^{\mathcal{N}}_{r(\phi)} = \{(v_i, v_{i+1}) \,|\, i < M \wedge (v_i, v_{i+1}) \notin E^{\mathcal{L}}_{r(\phi)}\}$$

Note that the edges $E^{\mathcal{L}}_{r(\phi)}$ pertaining to *literals* take precedence over edges $E^{\mathcal{N}}_{r(\phi)}$ created for *neighboring* request nodes. □

Note that the above definition only defined the request topology and did not specify demands or other restrictions. These will be set in the respective reductions. Matching the general construction of the request graph, the substrate graph is analogously defined, albeit introducing up to 7 substrate nodes per clause: the respective substrate nodes correspond to the *satisfying assignments* of the respective clause.

Definition 18 (Substrate Graph $G_{S(\phi)}$)**.** For a given 3-SAT formula ϕ the substrate graph $G_{S(\phi)} = (V_{S(\phi)}, E_{S(\phi)})$ is defined as follows. For each clause $\mathcal{C}_i \in \mathcal{C}_\phi$ and each potential assignment $a_{i,m} \in \mathcal{A}_i$ of truth values satisfying $\mathcal{C}_i$ a substrate node is used, i.e., $V_{S(\phi)} = \bigcup_{\mathcal{C}_i \in \mathcal{C}_\phi} \mathcal{A}_i$. Two substrate nodes $a_{i,m} \in V_{S(\phi)}$ and $a_{j,n} \in V_{S(\phi)}$ are connected in either of the following cases:

1) if $(v_i, v_j) \in E^{\mathcal{L}}_{r(\phi)}$ holds then the edge $(a_{i,m}, a_{j,n})$ is introduced if and only if the assignments $a_{i,m}$ and $a_{j,n}$ agree on the literals $\mathcal{L}_{i,j}$ contained in both clauses and
2) if $(v_i, v_j) \in E^{\mathcal{N}}_{r(\phi)}$ holds, then any edge $(a_{i,m}, a_{j,n})$ with $a_{i,m} \in \mathcal{A}_i$ and $a_{j,n} \in \mathcal{A}_j$ is introduced.

Accordingly, we set $E_{S(\phi)} = E^{\mathcal{L}}_{S(\phi)} \sqcup E^{\mathcal{N}}_{S(\phi)}$, with

$$E^{\mathcal{L}}_{S(\phi)} = \left\{ (a_{i,m}, a_{j,n}) \,\middle|\, \begin{array}{l} (v_i, v_j) \in E^{\mathcal{L}}_{r(\phi)} \text{ and} \\ a_{i,m}(x_l) = a_{j,n}(x_l) \text{ for } x_l \in \mathcal{L}_{i,j} \end{array} \right\}$$
$$E^{\mathcal{N}}_{S(\phi)} = \left\{ (a_{i,m}, a_{j,n}) \,\middle|\, (v_i, v_j) \in E^{\mathcal{N}}_{r(\phi)} \right\}. \qquad \square$$

C. The Base Lemma

In the following we give the base lemma, on which nearly all of our results are based. It shows the connection between the satisfiability of 3-SAT formulas and the existence of specific valid mappings introduced below.

Definition 19 (Satisfiable Valid Mappings $\mathcal{M}^{\mathrm{SAT}}_{r(\phi)}$)**.** We denote by $\mathcal{M}^{\mathrm{SAT}}_{r(\phi)}$ the set of valid mappings of $G_{r(\phi)}$ on $G_{S(\phi)}$, such that each virtual node v_i pertaining to the i-th clause is mapped on a substrate node of $\mathcal{A}_i$ and that each virtual edge is embedded using a single substrate edge, i.e.:

$$\mathcal{M}^{\mathrm{SAT}}_{r(\phi)} = \left\{ m \in \mathcal{M}_{r(\phi)} \,\middle|\, \begin{array}{l} m_V(v_i) \in \mathcal{A}_i \text{ for } v_i \in V_{r(\phi)} \text{ and} \\ |m_E(v_i, v_j)| = 1 \text{ for } (v_i, v_j) \in E_{r(\phi)} \end{array} \right\} \square$$

Lemma 20. *A 3-*SAT *formula ϕ is satisfiable iff. $\mathcal{M}^{\mathrm{SAT}}_{r(\phi)} \neq \emptyset$.*

Proof. We first show that if ϕ is satisfiable, then a mapping $m \in \mathcal{M}^{\mathrm{SAT}}_{r(\phi)}$ must exist. Afterwards, we show that if such a mapping $m \in \mathcal{M}^{\mathrm{SAT}}_{r(\phi)}$ exists, then ϕ must be satisfiable.

Assume that ϕ is satisfiable and let $\alpha : \mathcal{L}_\phi \to \{\mathrm{F}, \mathrm{T}\}$ denote an assignment of truth values, such that α satisfies ϕ. We construct a mapping $m = (m_V, m_E)$ for request $r(\phi)$ as follows. The virtual node $v_i \in V_{r(\phi)}$ corresponding to clause $\mathcal{C}_i$ is mapped onto the substrate node $a_{i,m} \in \mathcal{A}_i \subseteq V_{S(\phi)}$, iff. $a_{i,m}$ agrees with α on the assignment of truth values to the contained literals, i.e., $a_{i,m}(x_k) = \alpha(x_k)$ for $x_k \in \mathcal{C}_i$. As α satisfies ϕ, it satisfies each clause and hence $m_V(v_i) \in V_{S(\phi)}$ holds for all $\mathcal{C}_i \in \mathcal{C}_\phi$. The virtual edge $(v_i, v_j) \in E_{r(\phi)}$ is mapped via the direct edge between $m_V(v_i)$ and $m_V(v_j)$: if $(v_i, v_j) \in E^{\mathcal{L}}_{r(\phi)}$ holds, this edge $(m_V(v_i), m_V(v_j))$ must exist in $E^{\mathcal{L}}_{S}$, as $m_V(v_i) = a_{i,m}$ and $m_V(v_j) = a_{j,n}$ must agree by construction on the assignment of truth values for all literals. Secondly, if $(v_i, v_j) \in E^{\mathcal{N}}_{r(\phi)}$ holds, then the edge $(m_V(v_i), m_V(v_j))$ is always contained in $E^{\mathcal{N}}_{S}$. Hence, the constructed mapping m satisfies the conditions specified for $\mathcal{M}^{\mathrm{SAT}}_{r(\phi)}$, such that $m \in \mathcal{M}^{\mathrm{SAT}}_{r(\phi)}$ holds, hence completing the first half of the proof.

We now show that if there exists a mapping $m \in \mathcal{M}^{\mathrm{SAT}}_{r(\phi)}$ then the formula ϕ is indeed satisfiable. We constructively recover an assignment of truth values $\alpha : \mathcal{L}_\phi \to \{\mathrm{F}, \mathrm{T}\}$

from the mapping m by iteratively extending the initially empty assignment. Concretely, we iterate over the mappings of the virtual nodes corresponding to the clauses of $\mathcal{C}_\phi$ one by one according to the precedence relation of the clauses. By our assumption on the node mapping, $m_V(v_i) \in \mathcal{A}_i$ holds. Accordingly, as the substrate node $m_V(v_i)$ represents an assignment of truth values to the literals of clause $\mathcal{C}_i$, we extend α by setting $\alpha(x_k) \triangleq \big[m_V(v_i)\big](x_k)$ for all literals x_k contained in $\mathcal{C}_i$.

We first show that this extension is always valid in the sense that previously assigned truth values are never changed. To this end, assume that the clauses $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{i-1}$ were handled without any such violations. Hence, the literals $\bigcup_{j<i} \mathcal{L}_j$ have been assigned truth values in the first $i-1$ iterations not contradicting previous assignments. When extending α by the mapping of $m_V(v_i)$ in the i-th iteration, there are two cases to consider. First, if none of the literals $\mathcal{L}_i$ were previously assigned a truth value, i.e., if $\mathcal{L}_i \cap \bigcup_{j<i} \mathcal{L}_j = \emptyset$ holds, then the extension of α as described above cannot lead to a contradiction. Otherwise, if $\mathcal{L}_{i,\mathrm{pre}} = \mathcal{L}_i \cap \bigcup_{j<i} \mathcal{L}_j \neq \emptyset$ holds, we show that extending α by $m_V(v_i) = a_{i,m}$ does not change the truth value of any literal x_k contained in $\mathcal{L}_{i,\mathrm{pre}}$.

For the sake of contradiction, assume that $x_k \in \mathcal{C}_i$ is a literal, for which $\alpha(x_k)$ does not equal $\big[m_V(v_i)\big](x_k)$. As x_k was previously assigned a value, there must exist a clause $\mathcal{C}_j$ in which x_k was first used, such that $j < i$ holds. Let $m_V(v_i) = a_{i,m} \in \mathcal{A}_i$ and $m_V(v_j) = a_{j,n} \in \mathcal{A}_j$. By our assumption all edges are mapped using a single substrate edge, accordingly $m_E(v_i, v_j) = \langle (a_{j,n}, a_{i,m}) \rangle$ must hold. Hence, the substrate edge $(a_{j,n}, a_{i,m})$ must exist and must be contained in $E^{\mathcal{L}}_{r(\phi)}$ as $\mathcal{L}_{i,\mathrm{pre}} \neq \emptyset$ holds. Since $(v_j, v_i) \in E^{\mathcal{L}}_{r(\phi)}$ holds, the respective substrate edge must be contained in $E^{\mathcal{L}}_S$ by definition. As $E^{\mathcal{L}}_S$ contains only edges *if* assignments agree with each other, $\big[m_V(v_j)\big](x_k) = a_{j,n}(x_k) = a_{i,m}(x_k) = \big[m_V(v_i)\big](x_k)$ is obtained. This contradicts our assumption that $\alpha(x_k) \neq \big[m_V(v_i)\big](x_k)$ holds. Hence, the extension of α is always valid.

By construction of the substrate graph $G_{S(\phi)}$, the node set $\mathcal{A}_i \subseteq V_{S(\phi)}$ contains only the assignments of truth values for the literals $\mathcal{L}_i$ of clause $\mathcal{C}_i \in \mathcal{C}_\phi$ that satisfy the respective clause. Hence, the constructed assignment α satisfies each clause and thus the formula ϕ, hence concluding the proof. ■

The above base lemma is the heart of our reduction framework for obtaining our $\mathcal{NP}$-completeness and $\mathcal{NP}$-hardness results. The following formalizes this observation.

Lemma 21. *If the restrictions $\langle \mathbf{X} \,|\, \mathbf{Y} \rangle$ are sufficiently expressive to constrain feasible mappings of $G_{r(\phi)}$ to $G_{S(\phi)}$ to exactly the mappings of $\mathcal{M}^{\mathrm{SAT}}_{r(\phi)}$ for any 3-*SAT *formula ϕ, then the* D-VNEP $\langle \mathbf{X} \,|\, \mathbf{Y} \rangle$ *is $\mathcal{NP}$-complete.*

Proof. We outline the polynomial-time reduction from 3-SAT to the respective D-VNEP under constraints $\langle \mathbf{X} \,|\, \mathbf{Y} \rangle$. Considering any 3-SAT formula ϕ, the respective request and substrate graphs $G_{r(\phi)}$ and $G_{S(\phi)}$ are constructed. As the size of the request is bounded by the number of clauses and the size of the substrate is bounded by 7 times the number of clauses, the construction is polynomial. Under the assumption that the D-VNEP variant is sufficiently expressive to constrain the set of feasible embeddings to exactly $\mathcal{M}^{\mathrm{SAT}}_{r(\phi)}$, the question of whether the formula ϕ is satisfiable reduces to the question of whether a feasible embedding of $G_{r(\phi)}$ on $G_{S(\phi)}$ exists (cf. Lemma 20). This reduction yields the $\mathcal{NP}$-hardness of the respective D-VNEP variant under restrictions $\langle \mathbf{X} \,|\, \mathbf{Y} \rangle$ as 3-SAT is $\mathcal{NP}$-complete. As the Integer Program presented in Section II-D can be used to decide D-VNEP under any restrictions $\langle \mathbf{X} \,|\, \mathbf{Y} \rangle$ and solving Integer Programs lies in $\mathcal{NP}$ [24], the respective D-VNEP lies in $\mathcal{NP}$, hence showing the $\mathcal{NP}$-completeness of the respective D-VNEP variant. ■

IV. $\mathcal{NP}$-COMPLETENESS OF THE D-VNEP

Using the framework introduced above we now prove a series of $\mathcal{NP}$-completeness results for the D-VNEP. Specifically, we consider the D-VNEP under any combination of node and edge mapping restrictions. We first show the $\mathcal{NP}$-completeness of the capacitated D-VNEP variant $\langle \mathbf{VE} \,|\, \text{-} \rangle$ in the absence of additional restrictions. Given this result, we continue to prove the $\mathcal{NP}$-completeness of the other 5 combinations of restrictions. In particular, we even show the $\mathcal{NP}$-completeness for the two variants $\langle \text{-} \,|\, \mathbf{LN} \rangle$ and $\langle \text{-} \,|\, \mathbf{NR} \rangle$, which do not consider capacities at all. Hence, even when the physical network does not impose any resource constraints, finding embeddings satisfying node placement and latency or routing restrictions is already $\mathcal{NP}$-complete. Again, it must be noted that adding further restrictions only renders the respective D-VNEP harder, and hence the $\mathcal{NP}$-completeness is pertained when considering more than two restrictions (cf. Lemma 10).

Theorem 22. D-VNEP $\langle \mathbf{VE} \,|\, \text{-} \rangle$ *is $\mathcal{NP}$-complete.*

Proof. We show the statement via a polynomial-time reduction from 3-SAT according to Lemma 21. Specifically, we show how to constrain the set of feasible embeddings of $G_{r(\phi)}$ to $G_{S(\phi)}$ to exactly $\mathcal{M}^{\mathrm{SAT}}_{r(\phi)}$ (cf. Definition 19), such that each virtual node corresponding to the i-th clause is mapped on a substrate node corresponding to the i-th clause, albeit embedding all virtual edges using a single substrate edge.

To enforce the node mapping property of $\mathcal{M}^{\mathrm{SAT}}_{r(\phi)}$, namely that each virtual node $v_i \in V_{r(\phi)}$ must be mapped on nodes in $\mathcal{A}_i$, we set unit substrate node capacities and demands:

$$\begin{aligned} c_S(a_{i,m}) &= 1 \qquad \forall\, \mathcal{C}_i \in \mathcal{C}_\phi, a_{i,m} \in \mathcal{A}_i \\ c_{r(\phi)}(v_i) &= 1 \qquad \forall\, v_i \in V_{r(\phi)} \end{aligned}$$

Given these demands and capacities, we note the following. Firstly, as the substrate nodes have a capacity of 1 and the virtual nodes have a demand of 1, at most one virtual node may be placed on any substrate node. Secondly, the request graph $G_{r(\phi)}$ contains edges (v_i, v_{i+1}) for $i \in [N-1]$, while the substrate is acyclic with edges always being oriented towards substrate nodes pertaining to clauses with a higher index. Hence, if a virtual node $v_i \in V_{r(\phi)}$ is mapped on a substrate node $a_{k,o} \in V_{S(\phi)}$ corresponding to the k-th clause, then the virtual node v_{i+1} corresponding to the next clause must be mapped on a substrate node $a_{k',o'} \in V_{S(\phi)}$ with $k' \geq k+1$. Thus, if $v_i \in V_{r(\phi)}$ was to be mapped on a substrate node

$a_{k,o} \in V_{S(\phi)}$ with $k > i$, then at least the last node v_M of the chain $\langle v_i, v_{i+1}, \dots, v_M \rangle$ cannot be suitably mapped. By the same argument, $v_i \in V_{r(\phi)}$ cannot be mapped on a substrate node $a_{k,o}$ with $k < i$, as then at least the first node v_1 could not be mapped feasibly on any substrate node. Therefore, any feasible embedding must map the node $v_i \in V_{r(\phi)}$ on a substrate node $a_{i,m} \in V_{S(\phi)}$, therefore restricting the node mappings exactly as in the definition of the set $\mathcal{M}^{\text{SAT}}_{r(\phi)}$.

To enforce the edge mapping restrictions specified for $\mathcal{M}^{\text{SAT}}_{r(\phi)}$, namely that each virtual edge is embedded to exactly one substrate edge, the following non-unit edge capacities and demands are set for some λ with $0 < \lambda < 1/|\mathcal{C}_\phi|$:

$$\begin{aligned} c_S(e) &= 1 + \lambda \cdot j \qquad \forall \mathcal{C}_j \in \mathcal{C}_\phi, e \in \delta^-(\mathcal{A}_j) \\ c_{r(\phi)}(e) &= 1 + \lambda \cdot j \qquad \forall v_j \in V_{r(\phi)}, e \in \delta^-(v_j) \end{aligned}$$

Accordingly, the capacity of a substrate edge and the demand of a virtual edge is determined by the index of the clause its head is representing: the higher the clause-index of the edge's head, the higher the capacity. Given these capacities and demands, we now show that any virtual edge must be mapped on exactly one substrate edge. To this end, assume for the sake of contradiction that the virtual edge $(v_i, v_j) \in E_{r(\phi)}$ is not mapped on a single substrate edge. As v_i must be mapped on some node $a_{i,m} \in \mathcal{A}_i$ and v_j must be mapped on some node $a_{j,n} \in \mathcal{A}_j$, and as both the request and the substrate are directed acyclic graphs, the mapping of edge (v_i, v_j) must route through at least one intermediate node. Denote by $a_{k,l} \in \mathcal{A}_k$ for $i < k < j$ the first intermediate node lying on the path along which the edge (v_i, v_j) is routed. By construction, the capacity of the substrate edge $(a_{i,m}, a_{k,l})$ is $1 + \lambda \cdot k$. However, as $k < j$ holds and the edge (v_i, v_j) has a demand of $1 + \lambda \cdot j$, the edge (v_i, v_j) cannot be routed via $a_{k,l}$. Thus, the only feasible edges for embedding the respective virtual edges are the direct connections between any two substrate nodes.

Therefore, according to the above capacities and demands, any feasible embedding m satisfies the conditions of $\mathcal{M}^{\text{SAT}}_{r(\phi)}$ and is hence contained in it. On the other hand, the above imposed capacities do not constrain the set of feasible embeddings any further: each mapping $m \in \mathcal{M}^{\text{SAT}}_{r(\phi)}$ is a feasible embedding according to the above capacities. Thus, as the set of feasible embeddings equals $\mathcal{M}^{\text{SAT}}_{r(\phi)}$, by Lemma 21, any algorithm computing a feasible solution to the D-VNEP obeying node and edge capacities, can be used to decide 3-SAT, showing the $\mathcal{NP}$-completeness. ■

In the following the above $\mathcal{NP}$-completeness proof is adapted to other settings. All the proofs purely rely on constraining the set of feasible embeddings to the set $\mathcal{M}^{\text{SAT}}_{r(\phi)}$ and the application of Lemma 21. Hence, in the following we only prove that the set of feasible embeddings equals $\mathcal{M}^{\text{SAT}}_{r(\phi)}$.

Theorem 23. D-VNEP $\langle \mathbf{E} \,|\, \mathbf{N} \rangle$ *is* $\mathcal{NP}$*-complete.*

Proof. In this setting node placement restrictions and substrate edge capacities are enforced. Employing the node placement restrictions, we can force the mapping of virtual node $v_i \in V_{r(\phi)}$ onto substrate nodes $\mathcal{A}_i$ by setting $\overline{V}_S^{v_i} = V_{S(\phi)} \setminus \mathcal{A}_i$ for all $v_i \in V_{r(\phi)}$. Utilizing the same edge capacities and demands as in the proof of Theorem 22, virtual edges have to be mapped using a single edge, as intermediate nodes do not support the respective demand. Hence, the set of valid mappings is constrained to $\mathcal{M}^{\text{SAT}}_{r(\phi)}$ and the result follows. ■

Theorem 24. D-VNEP $\langle \mathbf{V} \,|\, \mathbf{R} \rangle$ *is* $\mathcal{NP}$*-complete.*

Proof. In this setting node capacities and routing restrictions must be obeyed. We employ the same node capacities as in the proof of Theorem 22, such that a virtual node $v_i \in V_{r(\phi)}$ may only be mapped on a substrate node contained in $\mathcal{A}_i$. Furthermore, routing restrictions are set to only allow direct edges. Specifically, for the virtual edge $(v_i, v_j) \in E_{r(\phi)}$ the set of forbidden edges $\overline{E}_S^{v_i, v_j}$ is set to $E_{S(\phi)} \setminus (\mathcal{A}_i \times \mathcal{A}_j)$. Hence, by using these restrictions, the set of valid mappings is constrained to exactly $\mathcal{M}^{\text{SAT}}_{r(\phi)}$ and the result follows. ■

Theorem 25. D-VNEP *is* $\mathcal{NP}$*-complete under restrictions* $\langle \text{-} \,|\, \mathbf{NR} \rangle$ *and* $\langle \text{-} \,|\, \mathbf{NL} \rangle$.

Proof. Both VNEP variants do not consider capacities. Allowing for node placement restrictions, the node mapping restrictions of $\mathcal{M}^{\text{SAT}}_{r(\phi)}$ are easily safeguarded (cf. proof of Theorem 23). By employing the same routing restrictions as in the proof of Theorem 24 the result follows directly for the case $\langle \text{-} \,|\, \mathbf{NR} \rangle$.

For $\langle \text{-} \,|\, \mathbf{NL} \rangle$, latency restrictions can be employed to enforce that virtual edges span at most a single substrate edge. Concretely, we set unit substrate edge latencies and unit virtual edge latency bounds:

$$\begin{aligned} l_S(e) &= 1 \quad \forall e \in E_{S(\phi)} \\ l_{r(\phi)}(e) &= 1 \quad \forall e \in E_{r(\phi)} \end{aligned}$$

Accordingly, each virtual edge can only be realized by using at most a single substrate edge. Furthermore, given the node mapping restrictions, the virtual nodes cannot be mapped onto the same substrate node. Hence, the set of feasible embeddings equals exactly $\mathcal{M}^{\text{SAT}}_{r(\phi)}$ and the result follows for $\langle \text{-} \,|\, \mathbf{NL} \rangle$. ■

Theorem 26. D-VNEP $\langle \mathbf{V} \,|\, \mathbf{L} \rangle$ *is* $\mathcal{NP}$*-complete.*

Proof. This variant enforces node capacities while also obeying latency restrictions. Again, utilizing unit node capacities and demands as in the proof of Theorem 22, the node mapping restrictions of $\mathcal{M}^{\text{SAT}}_{r(\phi)}$ are safeguarded. By employing unit substrate latencies and unit latency restrictions for each virtual edge as in the proof of Theorem 25, also the edge mapping restrictions of $\mathcal{M}^{\text{SAT}}_{r(\phi)}$ are enforced, thereby yielding the result. ■

V. $\mathcal{NP}$-Completeness of Computing Approximate Embeddings

Given the $\mathcal{NP}$-completeness results presented in Section IV, the question arises to which extent the hardness can be overcome when *relaxing constraints*, such as capacity or latency bounds, by considering *approximate* embeddings (cf. Definition 13). We first derive $\mathcal{NP}$-completeness results for computing α-approximate embeddings allowing node capacity violations and γ-approximate embeddings allowing latency violations for $\alpha < 2$ and $\gamma < 2$. For

β-approximate embeddings, another type of hardness result is obtained. We show a reduction from a variant of the edge-disjoint paths problem and show that β-approximate embeddings can in general not be computed in polynomial-time for $\beta \in \Theta(\log |V_S| / \log\log |V_S|)$.

Theorem 27. *Deciding* D-VNEP *under restrictions* $\langle \mathbf{VE} \,|\, \text{-} \rangle$, $\langle \mathbf{V} \,|\, \mathbf{R} \rangle$, *or* $\langle \mathbf{V} \,|\, \mathbf{L} \rangle$ *remains $\mathcal{NP}$-complete when asking only for α-approximate embeddings for any $\alpha < 2$.*

Proof. The $\mathcal{NP}$-completeness proofs under the restrictions $\langle \mathbf{VE} \,|\, \text{-} \rangle$, $\langle \mathbf{V} \,|\, \mathbf{R} \rangle$, and $\langle \mathbf{V} \,|\, \mathbf{L} \rangle$ relied all on the same argument to show that the virtual node $v_i \in V_{r(\phi)}$ must be mapped on substrate nodes contained in $\mathcal{A}_i \subseteq V_S$ (cf. Theorems 22, 24, and 26): due to the unit substrate node capacities and the unit node demands only a single virtual node can mapped on a substrate node and accordingly the chain of virtual nodes $v_1, v_2, \ldots, v_M$ must be embedded linearly using substrate nod sets $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_M$. Notably, the remaining parts of the proofs were only relying on the condition that $v_i \in V_{r(\phi)}$ must be mapped on a node in $\mathcal{A}_i \subseteq V_S$.

Clearly, in the above proofs one may increase the substrate node capacities by any factor $1 \leq \alpha < 2$ without any changes in the respective proofs: even if substrate nodes were to have a capacity of α, only a single virtual node can be hosted by any of the substrate nodes. As asking for an α-approximate embeddings is equivalent to finding a non-approximate embedding while increasing all node capacities by the factor α, any algorithm for the D-VNEP returning α-approximate embeddings for $\alpha < 2$ could still be used to decide 3-SAT and therefore even deciding whether an α-approximate embedding exists remains $\mathcal{NP}$-complete for any $\alpha < 2$. ∎

The $\mathcal{NP}$-completeness of deciding whether a γ-approximate embeddings exists uses the same argument:

Theorem 28. *Deciding* D-VNEP *under restrictions* $\langle \text{-} \,|\, \mathbf{NL} \rangle$ *and* $\langle \mathbf{V} \,|\, \mathbf{L} \rangle$ *remains $\mathcal{NP}$-complete when asking only for γ-approximate embeddings for any $\gamma < 2$.*

Proof. The proofs of Theorems 25 and 26 relied on the fact that due to the latency constraints each virtual edge must be mapped on a single substrate edge. As the latencies of substrate edges are uniformly set to 1 and all latency bounds are 1 as well, computing a γ-approximate embedding for $\gamma < 2$ implies that each virtual edge can still only be mapped on a single substrate edge. Analogously to the proof of Theorem 27, the respective D-VNEP variants remain $\mathcal{NP}$-complete even when only asking to decide whether a γ-approximate embedding exists. ∎

For β-approximate embeddings we employ a hardness result pertaining to a variant of the edge-disjoint paths problem, introduced below.

Definition 29 (DIREDPWC [17])**.** The decision variant of the Directed Edge-Disjoint Paths Problem with Congestion (DIREDPWC) is defined as follows. Given is a directed graph $G = (V, E)$ together with a set of $l \in \mathbb{N}$ source-sink pairs (commodities) $\{(s_k, t_k)\}_{k \in [l]}$, $s_k, t_k \in V$, and a constant $c \in \mathbb{N}$. The task is to decide whether for each commodity $k \in [l]$ a path P_k connecting s_k to t_k exists, such that at most c many paths are routed via any edge $e \in E$. □

It is well-known that DIREDPWC is hard to solve even when relaxing the congestion constraints:

Theorem 30 (Chuzhoy et al. [17])**.** *Let $n = |V|$ denote the number of nodes. Given an instance of the* DIREDPWC*, it is impossible to distinguish between the following two cases in polynomial-time, unless $\mathcal{NP} \subseteq \mathcal{BP}$-TIME$(\bigcup_{d \geq 1} n^{d \log\log n})$:*

1) *A solution with congestion $c = 1$ exists .*
2) *No solution with congestion $c \in \Theta(\log n / \log\log n)$ exists.*

Above, $\mathcal{BP}$-TIME$(f(n))$ denotes the class of problems solvable by probabilistic Turing machines in time $f(n)$ with bounded error-probability [26].

To apply this result for the DIREDPWC in the context of β-approximate embeddings, we give reductions from DIREDPWC to the D-VNEP variants $\langle \mathbf{E} \,|\, \mathbf{N} \rangle$ and $\langle \mathbf{VE} \,|\, \text{-} \rangle$. Importantly, these reductions are preserved when relaxing edge capacities, such that γ-approximate embeddings translate to solutions of the DIREDPWC with a congestion increase by a factor γ. We first give our reduction from DIREDPWC to D-VNEP under restrictions $\langle \mathbf{E} \,|\, \mathbf{N} \rangle$.

Theorem 31. *Solving the β-approximate* D-VNEP *under restrictions* $\langle \mathbf{E} \,|\, \mathbf{N} \rangle$ *is not possible in polynomial-time for $\beta \in \Theta(\log n / \log\log n)$ with $n = |V_S|$, unless $\mathcal{NP} \subseteq \mathcal{BP}$-TIME$(\bigcup_{d \geq 1} n^{d \log\log n})$ holds.*

Proof. Given a DIREDPWC instance on the graph $G = (V, E)$ with commodities $(s_k, t_k)_{k \in [l]}$ and congestion c, an equivalent D-VNEP instance consisting of the substrate graph $G_{S(\text{dir})} = (V_{S(\text{dir})}, E_{S(\text{dir})})$ and the request graph $G_{r(\text{dir})} = (V_{r(\text{dir})}, E_{r(\text{dir})})$ is constructed. Firstly, the substrate is set to equal the original graph, i.e., $G_{S(\text{dir})} = G$, and the request graph $G_{r(\text{dir})} = (V_{r(\text{dir})}, E_{r(\text{dir})})$ is defined as follows. $V_{r(\text{dir})}$ consists of two virtual nodes per commodity, $V_{r(\text{dir})} = \{i_k, j_k | k \in [l]\}$, and we set $E_{r(\text{dir})} = \{(i_k, j_k) | k \in [l]\}$. Let $\sigma : V_{r(\text{dir})} \to V_{S(\text{dir})}$ denote the function indicating the original substrate node locations of the respective commodities. Specifically, $\sigma(i_k) = s_k$ and $\sigma(j_k) = t_k$ holds for all $k \in [l]$. We employ node mapping requirements to force the mapping of virtual nodes i_k and j_k to the locations of the respective commodities s_k and t_k by setting $\overline{V}_S^i = V \setminus \{\sigma(i)\}$ for $i \in V_{r(\text{dir})}$. Setting edge capacities in the substrate to c (the congestion value) and virtual edge demands to 1, deciding the respective D-VNEP problem is equivalent to deciding DIREDPWC: any embedding $m = (m_V, m_E)$ of the D-VNEP instance induces a solution to the DIREDPWC instance by setting $P_k = m_E(i_k, j_k)$ for $k \in [l]$ and vice versa. Importantly, note that when considering β-approximate solutions for the above D-VNEP instance, a respective DIREDPWC solution of congestion $\beta \cdot c$ can be obtained in exactly the same way.

Given that β-approximate embeddings yield an increase in the congestion of the DIREDPWC solution by the same factor, we can now prove the impossibility to decide whether β-approximate embeddings exist for $\beta \in \Theta(\log |V_S| / \log\log |V_S|)$. For the sake of contradiction

assume that there exists a polynomial-time algorithm for the β-approximate D-VNEP for some $\beta \in o(\log |V_S|/\log\log |V_S|)$ and that $\mathcal{NP} \not\subseteq \mathcal{BP}\text{-}TIME(\bigcup_{d\geq 1} n^{d\log\log n})$ holds. Now, consider a DIREDPWC instance with $c = 1$ and assume that a feasible solution of congestion 1 exists. Clearly, the DIREDPWC solution of congestion $c = 1$ induces a feasible embedding (without exceeding edge capacities) of the respective D-VNEP instance. Accordingly, the β-approximate D-VNEP algorithm must return a β-approximate embedding in polynomial-time. The β-approximate solution can then be used to recover a solution to the original DIREDPWC instance having congestion $c = \beta \in o(\log |V_S|/\log\log |V_S|) = o(\log n/\log\log n)$, where $n = |V| = |V_S|$ denotes the number of nodes of the original DIREDPWC instance. However, under the assumption that $\mathcal{NP} \not\subseteq \mathcal{BP}\text{-}TIME(\bigcup_{d\geq 1} n^{d\log\log n})$ holds, the construction of this DIREDPWC instance of congestion $c = \beta$ contradicts Theorem 30, which states that such an approximate solution cannot (always) be found in polynomial-time. Hence, finding β-approximate in polynomial-time is in general impossible for some $\beta \in \Theta(\log n/\log\log n)$. ■

We will now derive a similar result for the impossibility of solving the β-approximate D-VNEP variant under capacity restrictions $\langle \mathbf{VE} \,|\, \text{-} \rangle$. In contrast to the variant $\langle \mathbf{E} \,|\, \mathbf{N} \rangle$, the reduction from DIREDPWC is slightly more involved as the endpoints of the commodities have to be fixed using node capacities only. To concisely state the result, we introduce the notion of core nodes. Specifically, considering a graph $G = (V, E)$ we use $V^c = \{u \in V \,|\, |\delta^+(u) \cup \delta^-(u)| \geq 2\}$ to denote the *core* nodes having more than a single incoming or outgoing edge. We may assume that any DIREDPWC instance does only consist of core nodes, as the following lemma shows.

Lemma 32. *Any* DIREDPWC *instance on the graph $G = (V, E)$ can be reduced to an equivalent instance on a graph $G_p = (V_p, E_p)$ with $V_p \subseteq V$, such that V_p contains only core nodes, i.e., $V_p^c = V_p$ holds.*

Proof. Given the initial graph $G = (V, E)$, the idea is that any non-core node $u \in V \setminus V^c$ does not offer any routing decisions and can therefore be removed from the instance. Specifically, consider a node $u \in V$ only having a single incoming edge e^- or only one outgoing edge e^+: the edge e^- will only be used by commodities whose target was mapped on u while the edge e^+ will only be used by commodities whose source was mapped on u. Furthermore, these commodities have to use the respective edges. When the number of commodities using such an edge lies above the congestion c, then clearly no solution can exist, while otherwise the respective node u can be removed, while reassigning the source or the target of the respective commodities from u to the node incident to u. By iterating this process, equivalent DIREDPWC instances are obtained until no non-core nodes exist anymore and the graph $G_p = (V_p, E_p)$ is obtained with $V_p = V_p^c$. ■

Theorem 33. *Solving the β-approximate* D-VNEP *under restrictions $\langle \mathbf{VE} \,|\, \text{-} \rangle$ is not possible in polynomial-time for $\beta \in \Theta(\log n/\log\log n)$ with $n = |V_S^c|$, unless $\mathcal{NP} \subseteq \mathcal{BP}\text{-}TIME(\bigcup_{d\geq 1} n^{d\log\log n})$ holds.*

Proof. We essentially use the same argumentation as in the proof of Theorem 31, but employ a different reduction from DIREDPWC to D-VNEP to fix the source and target mappings of commodities. Our reduction again takes as input a DIREDPWC instance on the graph $G = (V, E)$ with $l \in \mathbb{N}$ commodities $(s_k, t_k)_{k\in[l]}$ and outputs an equivalent D-VNEP instance consisting of a substrate $G_{S(\text{dir})}$ and a request $G_{r(\text{dir})}$.

To construct the D-VNEP instance, we first introduce some additional notation. We denote by $O^+ : V \to \mathbb{N}$ and $O^- : V \to \mathbb{N}$ the function that counts the number of times a node $v \in V$ occurs as source or as sink in the commodities, i.e., $O^+(v) = \sum_{k\in[l]:s_k=v} 1$ and $O^-(v) = \sum_{k\in[l]:t_k=v} 1$. To construct the substrate graph $G_{S(\text{dir})}$ the original graph G is extended as follows. For each node $v \in V$, we add $O^+(v)$ many copies $V^{+,v}_{S(\text{dir})} = \{v_1^+, v_2^+, \ldots, v^+_{O^+(v)}\}$ and $O^-(v)$ many copies $V^{-,v}_{S(\text{dir})} = \{v_1^-, v_2^-, \ldots, v^-_{O^-(v)}\}$. For each copy v_k^+ an edge (v_k^+, v) is added while for any sink node v_k^- the edge (v, v_k^-) is introduced. Additionally using the function $U : V \to [|V|]$ assigning each vertex a unique numeric identifier, we define substrate node capacities according to the following rule: all original nodes, $v \in V_{S(\text{dir})} \cap V$, are assigned a capacity of 0, while setting $c_S(v_k^+) = U(v)$ and $c_S(v_l^-) = U(v)$ for $v \in V$ and $k \in [O^+(v)]$ and $l \in [O^-(v)]$.

The request graph $G_{r(\text{dir})} = (V_{r(\text{dir})}, E_{r(\text{dir})})$ is constructed as in the proof of Theorem 31: $V_{r(\text{dir})}$ consists of two virtual nodes per commodity, $V_{r(\text{dir})} = \{i_k, j_k | k \in [l]\}$, and we set $E_{r(\text{dir})} = \{(i_k, j_k) | k \in [l]\}$. Using again $\sigma : V_{r(\text{dir})} \to V_{S(\text{dir})}$ to denote the function indicating the original substrate node location of the respective virtual nodes, the demand of virtual nodes is set to match the capacity of the respective endpoints they are to be mapped on: $c_{r(\text{dir})}(i) = U(\sigma(i))$ is set for $i \in V_{r(\text{dir})}$. Given these capacities, we now prove that for any feasible embedding $m = (m_V, m_E)$ any virtual node $i \in V_{r(\text{dir})}$ must indeed be mapped on a copy corresponding to $\sigma(i)$. Specifically, we show that for $i_k \in V_{r(\text{dir})}$ and $j_k \in V_{r(\text{dir})}$, corresponding to the source and the target of commodity k, $m_V(i_k) \in V^{+,\sigma(i_k)}_{S(\text{dir})}$ and $m_V(j_k) \in V^{+,\sigma(j_k)}_{S(\text{dir})}$ must hold.

We show the above statement by using an inductive argument and start off by first considering only the mappings of virtual nodes which shall be mapped on the (unique) substrate node $u \in V$ of highest value $U(u)$. Clearly, any virtual node $i_k \in V_{r(\text{dir})}$ with $\sigma(i_k) = u$ or any virtual node j_k with $\sigma(j_k) = u$ can only be mapped on substrate nodes of capacity $U(u)$. As only the substrate nodes contained in $V^{+,u}_{r(\text{dir})} \cup V^{-,u}_{r(\text{dir})}$ offer this capacity, $m_V(i_k) \in V^{+,u}_{S(\text{dir})} \cup V^{-,u}_{S(\text{dir})}$ and $m_V(j_k) \in V^{+,u}_{S(\text{dir})} \cup V^{-,u}_{S(\text{dir})}$ must hold. Furthermore, as the virtual node i_k induces a flow towards j_k and not both virtual nodes can be placed on the same copy of u, the virtual node must be mapped on a node contained in $V^{+,u}_{S(\text{dir})}$ as only these have outgoing edges. Analogously, the virtual node j_k must be mapped on a node contained in $V^{-,u}_{S(\text{dir})}$, as only these nodes have an incoming edge. As the demanded capacities and the substrate capacities of node u are equal, all virtual nodes can be mapped, while having to use all available capacities on the copies of node u. The above proof scheme can now be iteratively applied for substrate nodes offering

the second most capacity etc., thus concluding our proof that $m_V(i_k) \in V^{+,\sigma(i_k)}_{S(\text{dir})}$ and $m_V(j_k) \in V^{+,\sigma(j_k)}_{S(\text{dir})}$ holds for any $i_k, j_k \in V_{r(\text{dir})}$.

We now argue that by the above construction a solution to the DIREDPWC instance exists if and only if a solution to the respective D-VNEP instance exists. To this end, we set the edge capacities to the congestion value c for original edges contained in E and to 1 to any newly introduced edges towards copies of substrate nodes. Clearly, any embedding of the virtual edge $(i_k, j_k) \in E_{r(\text{dir})}$ must start at $m_V(i_k) \in V^{+,\sigma(i_k)}_{S(\text{dir})}$ and therefore must traverse $\sigma(i_k) = s_k$ as the second node. Analogously, the virtual edge must end in $m_V(j_k) \in V^{-,\sigma(j_k)}_{S(\text{dir})}$ and therefore must traverse $\sigma(j_k) = t_k$ as second last node. By the same argument any solution to the DIREDPWC instance yields a solution to the D-VNEP: considering commodity $k \in [l]$ first suitable (unused) source and target nodes $v^+ \in V^{+,s_k}_{S(\text{dir})}$ and $v^- \in V^{-,t_k}_{S(\text{dir})}$ are chosen and afterwards the embedding is extended by $m_V(i_k) = v^+$, $m_V(j_k) = v^-$, and $m_E(i_k, j_k) = \langle (v^+, s_k), P_k, (t_k, v^-) \rangle$.

Similar to the proof of Theorem 31 it remains to show that the above equivalence is preserved when considering β-approximate embeddings, which may exceed edge capacities by the factor β. This indeed remains true, as the above argument that the endpoints of each virtual edge $(i_k, j_k) \in E_{r(\text{dir})}$ must correctly be mapped on any of the respective source nodes $V^{+,s_k}_{S(\text{dir})}$ and sink nodes $V^{-,t_k}_{S(\text{dir})}$ only relied on (i) the chosen *node* capacities and demands and (ii) the orientation of the respective incident edges. Thus, a β-approximate embedding increases the congestion of the corresponding DIREDPWC solution by a factor of exactly β.

Lastly, using the same argument as in the proof of Theorem 31, assuming the existence of a DIREDPWC instance having a solution with congestion $c = 1$, it is impossible for any algorithm to find a β-approximate embedding with $\beta \in o(\log n / \log\log n)$ with $n = |V_{S(\text{dir})}|$, unless $\mathcal{NP} \subseteq \mathcal{BP}\text{-}TIME(\bigcup_{d\geq 1} n^{d \log\log n})$ holds. Notably, $V_{S(\text{dir})}$ contains the additional nodes $V^{+,v}_{S(\text{dir})}$ and $V^{-,v}_{S(\text{dir})}$, $v \in V$, introduced for the reduction. However, by Lemma 32 we may exclude these additional non-core nodes, hence strengthening the result to hold for $n = |V| = |V^c_{S(\text{dir})}|$. ■

VI. $\mathcal{NP}$-COMPLETENESS UNDER GRAPH RESTRICTIONS

Given the hardness of computing even approximate D-VNEP solutions, we now turn towards another type of problem relaxation, namely request graph restrictions. This is motivated as, e.g., Virtual Clusters, an undirected star network, can be optimally embedded in polynomial time [7] and efficient approximations for service chains are known [14]. We show that the D-VNEP remains $\mathcal{NP}$-complete when restricting requests to planar and degree-bounded graphs. Before showing this, we make the observation that acyclicity does not render the D-VNEP simpler.

Observation 34. Theorems 22 - 28 still hold when restricting the request and the substrate to acyclic graphs, as the constructed substrate and request graphs are acyclic by construction (cf. Definitions 17 and 18).

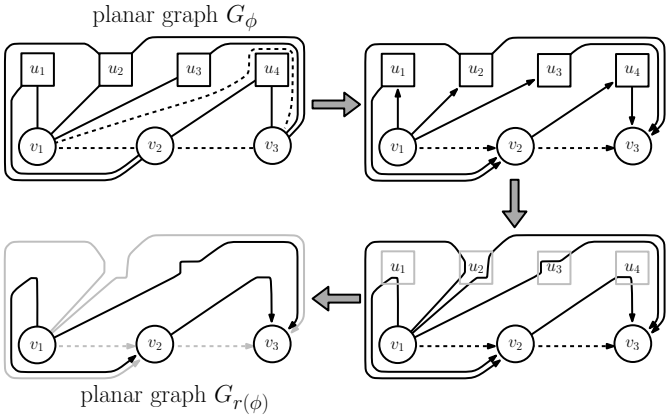

Fig. 3. Depicted is the transformation process of a planar graph G_ϕ (cf. Theorem 35) to the planar graph $G_{r(\phi)}$. Concretely, the example formula of Figure 2 is revisited, i.e., $\phi = C_1 \wedge C_2 \wedge C_3$, with $C_1 = x_1 \vee x_2 \vee x_3$, $C_2 = \bar{x}_1 \vee x_2 \vee x_4$, and $C_3 = x_2 \vee \bar{x}_3 \vee x_4$. The solid edges connect clause nodes and literal nodes, while the dashed edges link the clause nodes.
In the first step the edge $\{v_3, v_1\}$ is removed and all remaining edges are directed: edges between clause nodes and literal nodes are oriented towards literal nodes iff. the literal occurs in the respective clause for the first time (according to the ordering of clause nodes) and edges between clause nodes are oriented towards the clause with the higher index. In the second step, each outgoing edge of a literal node is joined with the single incoming edge (duplicating it when necessary), hence allowing to remove the literal nodes. In the last step, duplicate edges are removed, yielding the request graph $G_{r(\phi)}$. Each step of this transformation process safeguards the graph's planarity.

To obtain that the D-VNEP is $\mathcal{NP}$-complete for planar and degree-bounded request graphs, we consider reductions from a *planar* variant of 3-SAT, namely Clause-Linked Planar 3-Bounded 3-SAT (CP3B-3-SAT):

Theorem 35 (CP3B-3-SAT is $\mathcal{NP}$-complete [27])**.**
*Deciding the satisfiability of a 3-*SAT *formula remains $\mathcal{NP}$-complete under the following additional restrictions.*

1) *The undirected graph $G_\phi = (V_\phi, E_\phi)$ is planar, where*

$$V_\phi = \{v_i \,|\, C_i \in \mathcal{C}_\phi\} \cup \{u_k \,|\, x_k \in \mathcal{L}_\phi\}$$
$$E_\phi = \{\{v_i, u_k\} \,|\, \mathcal{C}_i \in \mathcal{C}_\phi, x_k \in x_k : x_k \in \mathcal{C}_i\} \cup \{\{v_i, v_{i+1} \,|\, i \in [M-1]\} \cup \{\{v_M, v_1\}\}\,.$$

2) *Each clause $\mathcal{C}_i \in \mathcal{C}_\phi$ contains at most three literals.*
3) *Each variable $x_k \in \mathcal{L}_\phi$ occurs in exactly three clauses.*

An example of a graph G_ϕ pertaining to a formula ϕ is depicted in Figure 3.

The following lemma connects CP3B-3-SAT formulas ϕ with the corresponding request graphs $G_{r(\phi)}$.

Lemma 36. *Given a* CP3B-3-SAT *formula ϕ, the following holds for the request graph $G_{r(\phi)}$ (cf. Definition 17):*

1) *The request graph $G_{r(\phi)}$ is planar.*
2) *The node-degree of $G_{r(\phi)}$ is bounded by 8.*

Proof. We consider an arbitrary CP3B-3-SAT formula ϕ to which the conditions of Theorem 35 apply. We first show that the corresponding request graph $G_{r(\phi)}$ is planar by detailing a transformation process leading from the planar graph G_ϕ to $G_{r(\phi)}$ while preserving planarity (see Figure 3).

Starting with the undirected graph G_ϕ, first the edge $\{v_M, v_1\}$ is removed and all edges are oriented: an edge

between a clause node and a variable node is oriented from a clause node to a literal node iff. the literal occurs in the respective clause for the first time according to the clauses' ordering. The edges between clause nodes are always oriented towards the clause with the higher index.

Given this directed graph, the literal nodes are now removed by joining the single incoming edge of a literal node with *each* of its outgoing edges. In particular, considering the literal node u_2 of Figure 3, the single incoming edge (v_1, u_2) is joined with the outgoing edges (u_2, v_2) and (u_2, v_3) to obtain the edges (v_1, v_2) and (v_1, v_3). As the duplication of the single incoming edge cannot refute planarity and all incoming and outgoing edges connect to the same node, the planarity of the graph is preserved in this step. Lastly, duplicate edges are removed to obtain the graph $G_{r(\phi)}$, which is, in turn, planar.

It remains to show, that the request graph $G_{r(\phi)}$ corresponding to ϕ exhibits a bounded node-degree of 8 (in the undirected interpretation of the graph $G_{r(\phi)}$). To see this, we note the following. Based on the second and third conditions of CP3B-3-SAT (cf. Theorem 35) each clause node is connected to at most two different other clauses via each literal node it is connected to, yielding at most 6 neighbors. Furthermore, any clause is directly connected to at most two further clause nodes via the edges between clause nodes, yielding a node-degree bound of 8. ■

Given the above, we derive the following theorem:

Theorem 37. *Theorems 22 - 28 hold when restricting the request graphs to be planar and degree 8-bounded. Theorems 31 and 33 hold for planar and degree 1-bounded graphs.*

Proof. Our $\mathcal{NP}$-completeness proofs in Section IV and Section V (except for Theorems 31 and 33) relied solely on the reduction from 3-SAT to D-VNEP using the base Lemma 20. As formulas of CP3B-3-SAT are a strict subset of the 3-SAT formulas, the base Lemma 20 is still applicable for CP3B-3-SAT formulas. However, due to the structure of CP3B-3-SAT formulas, the corresponding requests in the reductions are planar and exhibit a node-degree bound of 8 by Lemma 36. Hence, solving the D-VNEP is $\mathcal{NP}$-complete, even when restricting the requests to planar and / or degree-bounded ones. Lastly, we note that Theorems 31 and 33 hold for planar and degree 1-bounded request graphs, as in the reduction only such requests were considered. ■

VII. Hardness of O-VNEP and Discussion of Practical Implications

In this section the results obtained for the D-VNEP are translated into hardness results for the O-VNEP *under any objective*. Furthermore, an additional hardness result pertaining to the computation of convex combinations of valid mappings is discussed. This result sheds light on the recently discovered first *parametrized*, i.e., non-polynomial, approximations of the offline O-VNEP for general request graphs [16]. We begin by showing that the hardness results for the D-VNEP translate to the ($\mathcal{NP}$-)hardness of the O-VNEP. Even more, we show that the O-VNEP is inapproximable under *any* objective.

Theorem 38. *(i) The* O-VNEP *is $\mathcal{NP}$-hard and inapproximable (unless $\mathcal{P}=\mathcal{NP}$) under the following restrictions:* $\langle$ **VE** | **-** $\rangle$, $\langle$ **E** | **N** $\rangle$, $\langle$ **V** | **R** $\rangle$, $\langle$ **-** | **NR** $\rangle$, $\langle$ **-** | **NL** $\rangle$, $\langle$ **V** | **L** $\rangle$.

(ii) The result (i) remains valid for acyclic substrates and acyclic, planar, degree 8-bounded requests and when allowing for α-, or γ-approximate embeddings for $\alpha < 2$ or $\beta < 2$.

(iii) Theorems 31 and 33 remain true for the O-VNEP *under any objective. Hence, computing any β-approximate* O-VNEP *solution within polynomial-time is impossible for $\beta \in \Theta(\log n / \log\log n)$ with $n = |V_S|$ for $\langle$ **E** | **N** $\rangle$ and $n = |V_S^c|$ for $\langle$ **VE** | **-** $\rangle$, unless $\mathcal{NP} \subseteq \mathcal{BP}$-TIME$(\bigcup_{d\geq 1} n^{d \log\log n})$.*

Proof. Consider an O-VNEP instance consisting of request graph G_r and substrate graph G_S and denote by $\mathcal{M}_r$ the set of all feasible embeddings. While the task of the respective D-VNEP instance is to decide whether $\mathcal{M}_r = \emptyset$ holds, the task of the O-VNEP is not only to decide whether there exists a feasible embedding but to return one optimizing the objective (cf. Definition 5). Hence, all above results pertaining to the $\mathcal{NP}$-completeness of the D-VNEP translate into the $\mathcal{NP}$-hardness of the O-VNEP by the same reduction framework used above. Specifically, any given 3-SAT formula ϕ is satisfiable if and only if the respective O-VNEP algorithm returns any (optimal) feasible embedding. Clearly, this reduction still works when allowing the O-VNEP algorithm to return any feasible solution only approximating the optimal objective, as any feasible solution is proof of the satisfiability of the formula ϕ. Hence, the O-VNEP is inapproximable under any objective in polynomial-time unless $\mathcal{P}=\mathcal{NP}$ holds. Hence, due to Theorems 22 to 26, the first statement holds.

The second statement is a direct corollary of our above observations and the Theorems 27, 28, and 37.

The third statement holds as the respective reductions used in Theorem 31 and 33 can be easily adapted to the O-VNEP: finding an optimal or approximate solution only exceeding edge capacities by a factor β encompasses finding such a β-approximate solution in the first place. Based on the hardness of DirEDPwC, this remains impossible unless $\mathcal{NP} \subseteq \mathcal{BP}$-TIME$(\bigcup_{d\geq 1} n^{d \log\log n})$. ■

Notably, the above theorem also holds for O-VNEP variants considering several requests instead of a single one: finding feasible embeddings for several requests is at least as hard as finding a feasible embedding for a single request.

In the following, the implications of our results are discussed in the light of recent approximation algorithms for an offline variant of the O-VNEP under the restrictions $\langle$ **VE** | **NR** $\rangle$ and $\langle$ **VE** | **NRL** $\rangle$ [15], [16]. Specifically, the studied offline O-VNEP variant asks for deciding which of the requests to feasibly embed to maximize the obtained profit, while not exceeding resource capacities. As discussed above, the O-VNEP is in general inapproximable. Hence, the authors of [15], [16] consider a relaxed model, allowing for resource augmentations both on the nodes and the edges. The obtained approximations use Linear Programming to compute optimal convex combinations of valid mappings for which the feasibility is not ensured. However, even computing valid mappings of minimal cost, introduced as Valid Mapping Problem (VMP)

in [16], is $\mathcal{NP}$-hard *and inapproximable* in polynomial-time under the restrictions $\langle \text{-} | \mathbf{NR} \rangle$ and $\langle \text{-} | \mathbf{NRL} \rangle$. While this rules out polynomial-time algorithms in general, it was shown in [16] that *parametrized* algorithms for the VMP exist, whose runtime is exponential in the treewidth of the request graph. As the treewidth of specific graph classes is bounded for example for outer-planar graphs, the respective approximations have a polynomial-runtime for these graph classes. Our results show that such parametrized algorithms are the best we can hope for. Specifically, by Theorem 38, solving the VMP or the O-VNEP is impossible in polynomial-time for planar request graphs, unless $\mathcal{P} = \mathcal{NP}$ holds.

VIII. Conclusion

This work has presented a comprehensive set of hardness results for several variants of the VNEP, which lie at the core of many resource allocation problems in networks. Our results are negative in nature: we show that the decision variants are $\mathcal{NP}$-complete and the respective optimization variants are $\mathcal{NP}$-hard and inapproximable (unless $\mathcal{P} = \mathcal{NP}$ holds). This remains true even when restricting request graphs to planar graphs and when relaxing constraints within certain bounds.

As these results are proven for any combination of node and edge mapping restrictions, our results are of general importance and apply also to (sub-)problems encountered for example in the contexts of service function chaining or network function virtualization.

References

[1] M. Rost and S. Schmid, "Charting the complexity landscape of virtual network embeddings," in *2018 IFIP Networking Conference (IFIP Networking)*, 5 2018, pp. 1–9.

[2] J. C. Mogul and L. Popa, "What we talk about when we talk about cloud network performance," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 5, pp. 44–48, Sep. 2012.

[3] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1888–1906, 4 2013.

[4] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, 10 2014, pp. 7–13.

[5] J. M. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture," RFC 7665, Oct. 2015.

[6] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *Proceedings of the ACM SIGCOMM 2011 Conference*, ser. SIGCOMM '11. New York, NY, USA: ACM, 2011, pp. 242–253.

[7] M. Rost, C. Fuerst, and S. Schmid, "Beyond the stars: Revisiting virtual cluster embeddings," *SIGCOMM Computer Communication Review (CCR)*, vol. 45, no. 3, pp. 12–18, Jul. 2015.

[8] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, 2012.

[9] G. Even, M. Medina, G. Schaffrath, and S. Schmid, "Competitive and deterministic embeddings of virtual networks," *Theoretical Computer Science*, vol. 496, pp. 184 – 194, 2013.

[10] M. Rost, S. Schmid, and A. Feldmann, "It's about time: On optimal virtual network embeddings under temporal flexibilities," in *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, 5 2014, pp. 17–26.

[11] G. Schaffrath, S. Schmid, and A. Feldmann, "Optimizing long-lived cloudnets with migrations," in *2012 IEEE Fifth International Conference on Utility and Cloud Computing*, 11 2012, pp. 99–106.

[12] L. R. Bays, R. R. Oliveira, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Security-aware optimal resource allocation for virtual network embedding," in *8th International Conference on Network and Service Management, CNSM 2012*, Oct 2012, pp. 378–384.

[13] J. Inführ and G. R. Raidl, "Introducing the virtual network mapping problem with delay, routing and location constraints," in *Network Optimization*, J. Pahl, T. Reiners, and S. Voß, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 105–117.

[14] G. Even, M. Rost, and S. Schmid, "An approximation algorithm for path computation and function placement in sdns," in *Structural Information and Communication Complexity*, J. Suomela, Ed. Cham: Springer International Publishing, 2016, pp. 374–390.

[15] M. Rost and S. Schmid, "Virtual network embedding approximations: Leveraging randomized rounding," *IEEE/ACM Transactions on Networking*, vol. 27, no. 5, pp. 2071–2084, Oct 2019.

[16] M. Rost, E. Döhne, and S. Schmid, "Parametrized complexity of virtual network embeddings: Dynamic & linear programming approximations," *SIGCOMM Comput. Commun. Rev.*, vol. 49, no. 1, pp. 3–10, Feb. 2019.

[17] J. Chuzhoy, V. Guruswami, S. Khanna, and K. Talwar, "Hardness of routing with congestion in directed graphs," in *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, ser. STOC '07. New York, NY, USA: ACM, 2007, pp. 165–178.

[18] J. Diaz, J. Petit, and M. Serna, "A survey of graph layout problems," *ACM Comput. Surv.*, vol. 34, no. 3, pp. 313–356, Sep. 2002.

[19] D. Eppstein, *Subgraph Isomorphism in Planar Graphs and Related Problems*, 2002, pp. 283–309.

[20] D. G. Andersen, "Theoretical approaches to node assignment," Dec. 2002, [Online]. Available: http://repository.cmu.edu/compsci/86/.

[21] E. Amaldi, S. Coniglio, A. M. Koster, and M. Tieves, "On the computational complexity of the virtual network embedding problem," *Electronic Notes in Discrete Mathematics*, vol. 52, pp. 213 – 220, 2016.

[22] S. Agarwal, S. Rajakrishnan, A. Narayan, R. Agarwal, D. Shmoys, and A. Vahdat, "Sincronia: Near-optimal network design for coflows," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '18. New York, NY, USA: ACM, 2018, pp. 16–29.

[23] O. Rottenstreich, I. Keslassy, Y. Revah, and A. Kadosh, "Minimizing delay in network function virtualization with shared pipelines," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 1, pp. 156–169, Jan 2017.

[24] C. H. Papadimitriou, "On the complexity of integer programming," *Journal of the ACM*, vol. 28, no. 4, 1981.

[25] R. M. Karp, *Reducibility among Combinatorial Problems*. Boston, MA: Springer US, 1972, pp. 85–103.

[26] S. Arora and B. Barak, *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[27] M. R. Fellows, J. Kratochvil, M. Middendorf, and F. Pfeiffer, "The complexity of induced minors and related problems," *Algorithmica*, vol. 13, no. 3, pp. 266–282, Mar 1995.

Matthias Rost is a postdoctoral researcher at Technische Universität Berlin, Germany, from which he also received his MSc (2014) and his PhD (2019). He was awarded the KuVS prize for his master thesis on computing virtual aggregation and multicast trees by the German Informatics Society. His main research focus is the theoretical design of provably good algorithms for service orchestration in networks and their application in practice.

Stefan Schmid is a Professor at the University of Vienna, Austria. He received his MSc (2004) and PhD (2008) from ETH Zurich, Switzerland. Subsequently, Stefan Schmid worked as postdoc at TU Munich and the University of Paderborn (2009). From 2009 to 2015, he was a senior research scientist at the Telekom Innovations Laboratories (T-Labs) in Berlin, Germany, and from 2015 to 2018 an Associate Professor at Aalborg University, Denmark. His research interests revolve around algorithmic problems of networked and distributed systems, currently with a focus on self-adjusting networks (related to his ERC project AdjustNet).