Charting the Complexity Landscape of Virtual Network Embeddings

Matthias Rost
Technische Universität Berlin
Email: mrost@inet.tu-berlin.de

Stefan Schmid
University of Vienna
Email: stefan_schmid@univie.ac.at

***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 (from where the problem originated), over the embedding of batch-processing workloads (virtual clusters) to the embedding of service function chains. The different applications come with their own specific requirements and constraints, including node mapping constraints, routing policies, and latency constraints. While the VNEP has been studied intensively over the last years, complexity results are only known for specific models and we lack a comprehensive understanding of its hardness.**

This paper charts the complexity landscape of the VNEP by providing a systematic analysis of the hardness of a wide range of VNEP variants, using a unifying and rigorous proof framework. In particular, we show that the problem of finding a feasible embedding is $\mathcal{NP}$-complete in general, and, hence, the VNEP cannot be approximated *under any objective*, unless $\mathcal{P} = \mathcal{NP}$ holds. Importantly, we derive $\mathcal{NP}$-completeness results also for finding approximate embeddings, which may violate, e.g., capacity constraints by certain factors. Lastly, we prove that our results still pertain when restricting the request graphs to planar or degree-bounded graphs.

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. Indeed, it has been shown that cloud application performance can suffer significantly from interference on the communication network [1].

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). Figure 1 depicts an example of embedding a request graph.

The problem is known in the networking community under the name *Virtual Network Embedding Problem* (VNEP) and

ISBN 978-3-903176-08-9

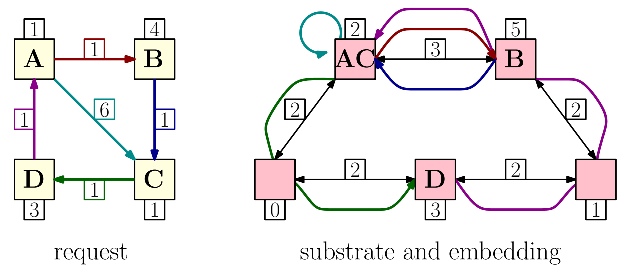

Fig. 1. Example request (left) together with an exemplary embedding on a substrate network (right). The numeric labels at the network elements denote the resource demands (of the request) and the available capacity (of the substrate), respectively. In the embedding, the single request edge $(\mathbf{C}, \mathbf{D})$ (green) is realized via a path of length two in the substrate. As request nodes $\mathbf{A}$ and $\mathbf{C}$ are collocated on the same substrate node, the edge $(\mathbf{A}, \mathbf{C})$ does not use any substrate edges and hence uses no edge resources. Note that the allocations induced by the embedding do not exceed the substrate's capacities.

has been studied intensively for over a decade [2], [3]. Besides the rather general study of the VNEP, which emerged originally from the study of testbed provisioning, essentially the same problems are considered in the context of Service Function Chaining [4], [5], as well as in the context of embeddings Virtual Clusters, a specific batch processing request abstraction [6], [7].

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 [2], [3], maximizing the profit by exerting admission control [8], [9], and minimizing the maximal load [4], [10].

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

- Restrictions on the placement of virtual nodes first arose to enforce closeness to locations of interest [2], but were also used in the context of privacy policies to restrict mappings to certain countries [11]. However, these restrictions are now also used in the context of Service Function Chaining, as specific functions may only be mapped on x86 servers, while firewall appliances cannot [4], [5].
- Routing restrictions first arose in the context of expressing security policies, as for example some traffic may not be

TABLE I
OVERVIEW ON RESULTS OBTAINED IN THIS PAPER.

VNEP variants		Identifier according to Definition 8	$\langle\, \mathbf{VE} \,\vert\, \textbf{-} \,\rangle$	$\langle\, \mathbf{E} \,\vert\, \mathbf{N} \,\rangle$	$\langle\, \mathbf{V} \,\vert\, \mathbf{R} \,\rangle$	$\langle\, \textbf{-} \,\vert\, \mathbf{NR} \,\rangle$	$\langle\, \textbf{-} \,\vert\, \mathbf{NL} \,\rangle$
		Enforcing Node Capacities	✓	⋆	✓	⋆	⋆
		Enforcing Edge Capacities	✓	✓	⋆	⋆	⋆
		Enforcing Node Placement Restrictions	⋆	✓	⋆	✓	✓
		Enforcing Edge Routing Restrictions	⋆	⋆	✓	✓	⋆
		Enforcing Latency Restrictions	⋆	⋆	⋆	⋆	✓
Results	Section IV	$\mathcal{NP}$-completeness and inapproximability under any objective	Thm. 19	Thm. 20	Thm. 21	Thm. 22	Thm. 22
	Section V	$\mathcal{NP}$-completeness and inapproximability when increasing *node capacities* by a factor $\alpha < 2$	Thm. 23	-	Thm. 23	-	-
		Inapproximability when increasing edge capacities by a factor $\beta \in \Theta(\log n / \log\log n)$ (unless $\mathcal{NP} \subseteq \mathcal{BP}$-$\mathit{TIME}(\bigcup_{d>1} n^{d \log\log n})$)	Thm. 25	Thm. 25	-	-	-
		$\mathcal{NP}$-completeness and inapproximability when loosening latency bounds by a factor $\gamma < 2$	-	-	-	-	Thm. 24
	Section VI	Results are preserved for acyclic substrates (except for Thm. 25)	Obs. 26				
		Results are preserved for acyclic, planar, degree-bounded requests	Thm. 31				

routed via insecure domains or physical links shall not be shared with other virtual networks [3], [12].

- Restrictions on latencies 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: Several dozens of algorithms were proposed to solve the VNEP and its siblings, including the Virtual Cluster Embedding [6] and Service Function Chain Embedding problem [3]. Most approaches to solve the VNEP either rely on heuristics [2] or metaheuristics [3]. On the other hand, several works study exact (non-polynomial time) algorithms to solve the problem to (near-)optimality or to devise heuristics. Mixed Integer Programming is the most widely used exact approach [4], [9], [13].

Only recently, approximation algorithms providing quality guarantees for the VNEP have been presented. In particular, the embedding of chains is approximated under assumptions on the requested resources and the achievable benefit in [14]. In [15] approximations for cactus request graphs are detailed, while [16] presents fixed-parameter tractable approximations for *arbitrary* request graph topologies.

c) Complexity Results: Surprisingly, despite the relevance of the problem and the large body of literature, the complexity of the underlying problems has not received much attention. While it can be easily seen that the Virtual Network Embedding Problem encompasses several $\mathcal{NP}$-hard problems as e.g. the k-disjoint paths problem [17], the minimum linear arrangment 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.

Bibliographic Note: An extended version of this paper was published as a technical report [22]. In addition to the contents found in this paper, it contains a detailed Integer Programming formulation able to solve the VNEP variants considered in this paper. Furthermore, it contains the proof of Theorem 25, that is omitted here due to space constraints.

B. Contributions and Overview

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

- We show the $\mathcal{NP}$-completeness of *five* different VNEP variants in Section IV. For example, we consider the variant only enforcing capacity constraints, but also one in which only node placement and latency restrictions must be obeyed *in the absence of capacity constraints*.
- We extend these results in Section V and show that the considered variants remain $\mathcal{NP}$-complete even when computing *approximate* embeddings, which may exceed latency or capacity constraints by certain factors.
- Lastly, we show in Section VI that the respective VNEP variants remain $\mathcal{NP}$-complete even when restricting substrate graphs to directed acyclic graphs (DAGs) and request graphs to planar, degree-bounded DAGs.

As we are proving $\mathcal{NP}$-completeness throughout this paper, the implications of our results are severe. Given the $\mathcal{NP}$-completeness of finding *any* feasible solution, finding an optimal solution *subject to any objective* is at least $\mathcal{NP}$-hard. Furthermore, unless $\mathcal{P} = \mathcal{NP}$ holds, the respective variants cannot be approximated to within *any* factor.

Table I summarizes our results and is to be read as follows. Any of the five rightmost columns represents a specific VNEP variant. The ✓ symbol indicates restrictions that are enforced, while the ⋆ symbol indicates restrictions which are not considered. Importantly, enabling a ⋆ restriction, does not change the results (cf. Lemma 9). Considering a specific variant, the respective column should be read from top to bottom. Considering for example $\langle\, \mathbf{VE} \,\vert\, \textbf{-} \,\rangle$, its $\mathcal{NP}$-completeness is shown in Theorem 19 while its inapproximability when relaxing node capacity constraints is shown in Theorem 23. Lastly, all results also hold under the graph restrictions of the two bottom rows.

II. Formal Model

We now formally introduce the VNEP and its variants.

Notation: The following notation is used throughout this work. We use $[x]$ to denote $\{1, 2, \dots, x\}$ 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 node $v \in V$. When considering functions on tuples, we omit the parantheses 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 CPUs while the capacity $c_S(u, v)$ of edge $(u, v) \in E_S$ represents the available bandwidth. By allowing to set substrate capacities to ∞, the capacity constraints on the respective substrate elements can be effectively disabled. We denote by $\mathcal{P}_S$ the set of all simple paths in G_S.

A request is similarly 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 task is to find a *mapping* of request graph G_r on the substrate network G_S, i.e. a mapping of request nodes to substrate nodes and a mapping of request edges 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 $V^i_S = \{u \in V_S | c_S(u) \geq c_r(i)\}$ the set of substrate nodes supporting the mapping of node $i \in V_r$ and by $E^{i,j}_S = \{(u, v) \in E_S | c_S(u, v) \geq c_r(i, j)\}$ 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 $m_V(i) \in V^i_S$ 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 simple paths in G_S connecting $m_V(i)$ to $m_V(j)$, such that $m_E(i, j) \subseteq E^{i,j}_S$ holds for $(i, j) \in E_r$. □

Considering the above definition, note the following. Firstly, the mapping $m_E(i, j)$ of the virtual edge $(i, j) \in E_r$ may be empty, if (and only if) i and j are mapped on the same substrate node. Secondly, the definition only enforces that single resource allocations do not exceed the available capacity. To enforce that the cumulative allocations respect capacities, we introduce the following:

Definition 2 (Allocations)**.** We denote by $A_m(x) \in \mathbb{R}_{\geq 0}$ the resource allocations induced by valid mapping $m = (m_V, m_E)$ on substrate element $x \in G_S$ and define

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

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 represents a feasible embedding, if the allocations do not exceed the capacity, i.e. $A_m(x) \leq c_S(x)$ holds for $x \in G_S$. □

In this paper we study the *decision* variant of the VNEP, asking whether there exists a feasible embedding:

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

B. *Variants of the* VNEP *& Nomenclature*

As discussed when reviewing the related work in Section I-A, additional requirements 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 5 (Node Placement Restrictions)**.** For each virtual node $i \in V_r$ a set of forbidden substrate nodes $\overline{V}^i_S \subset V_S$ is provided. Accordingly, the set of allowed nodes V^i_S is defined to be $\{u \in V_S \setminus \overline{V}^i_S \,|\, c_S(u) \geq c_r(i)\}$. □

Definition 6 (Routing Restrictions)**.** For each virtual edge $(i, j) \in E_r$ a set of forbidden substrate edges $\overline{E}^{i,j}_S \subseteq E_S$ is provided. Accordingly, the set of allowed edges $E^{i,j}_S$ is set to be $\{(u, v) \in E_S \setminus \overline{E}^{i,j}_S \,|\, c_S(u, v) \geq c_r(i, j)\}$. □

Definition 7 (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 $(i, j) \in E_r$, is 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 $(i, j) \in E_r$. □

We introduce the following taxonomy to denote the different problem variants.

Definition 8 (Taxonomy)**.** We use the notation $\langle\, \mathbf{C} \,|\, \mathbf{A} \,\rangle$ 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 used. When node or edge capacities are *not* considered, we assume the capacities of the respective substrate elements to be ∞.
- **A** Considering the additional restrictions, the abbreviations **-**, **N**, **L**, and **R** stand for no restrictions, node placement, latency, and routing restrictions, respectively. □

Accordingly, $\langle\, \mathbf{VE} \,|\, \textbf{-} \,\rangle$ indicates the classic VNEP without additional constraints while obeying capacities and $\langle\, \textbf{-} \,|\, \mathbf{NL} \,\rangle$ indicates the combination of node placement and latency restrictions while neither considering node or edge capacities. We note that the introduction of more restrictions only makes the respective problem harder:

Lemma 9. *A VNEP variant* $\langle\,\mathbf{A}\,|\,\mathbf{C}\,\rangle$ *that encompasses all restrictions of* $\langle\,\mathbf{A'}\,|\,\mathbf{C'}\,\rangle$ *is at least as hard as* $\langle\,\mathbf{A'}\,|\,\mathbf{C'}\,\rangle$.

Proof. The capacity constraints as well as the additional requirements are all formulated in such a fashion that any one of these can be disabled. Considering capacities and latencies, one may set the respective substrate capacities to ∞ and the latencies of substrate edges to 0, respectively. For node placement and edge restrictions one may set the forbidden node and edge sets to the empty set. Hence, a trivial reduction from $\langle\,\mathbf{A}\,|\,\mathbf{C}\,\rangle$ to $\langle\,\mathbf{A'}\,|\,\mathbf{C'}\,\rangle$ exists and the result follows. ■

C. Relaxing Constraints: Approximate Embeddings

Within this work, we show the VNEP to be $\mathcal{NP}$-complete under many meaningful restriction combinations. This in turn also implies the inapproximability of the respective VNEP variants (unless $\mathcal{P}=\mathcal{NP}$ holds). 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 a valid embedding exists that satisfies all capacity constraints, one might for example seek an embedding that uses at most two times the actual capacities. We refer to these embeddings as approximate embeddings:

Definition 10 (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 $\alpha, \beta, \gamma \geq 1$:

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

III. Reduction Framework

This section presents the main insight and contribution of our paper, namely a generic reduction framework that allows to derive hardness results by slightly tailoring the proof for the individual problem variants. Our reduction framework relies on 3-SAT and we first introduce some notation. Afterwards we continue by constructing a (partial) VNEP instance, whose solution will indicate whether the 3-SAT formula is satisfiable.

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 11 (Karp [23])**.** *Deciding 3-*SAT *is* $\mathcal{NP}$*-complete.*

For reducing 3-SAT to VNEP, it is important that the clauses be ordered and we define the following:

Definition 12 (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 $i' \in [i-1]$. □

As we are interested the satisfiability of a 3-SAT formula ϕ, we define the set of satisfying assignments *per clause*:

Definition 13 (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 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 embedding of $G_{r(\phi)}$ on $G_{S(\phi)}$ exists. Figure 2 illustrates the construction described in the following.

Definition 14 (Substrate Graph $G_{S(\phi)}$)**.** For a given 3-SAT formula ϕ we define the substrate graph $G_{S(\phi)} = (G_{S(\phi)}, E_{S(\phi)})$ as follows. For each clause $\mathcal{C}_i \in \mathcal{C}_\phi$ and each potential assignment of truth values satisfying $\mathcal{C}_i$, a substrate node is constructed, i.e. we set $V_{S(\phi)} = \bigcup_{\mathcal{C}_i\in\mathcal{C}_\phi} \mathcal{A}_i$. We connect two substrate nodes $a_{i,m} \in V_{S(\phi)}$ and $a_{j,n} \in V_{S(\phi)}$, iff. a literal x_k is introduced in the clause $\mathcal{C}_i$ for the first time and is also used in clause $\mathcal{C}_j$, *and* $a_{i,m}$ and $a_{j,n}$ agree on the truth values of the literals contained in both clauses. Formally, we set:

$$E_{S(\phi)} = \left\{ (a_{i,m}, a_{j,n}) \;\middle|\; \begin{array}{l} \exists x_k \in \mathcal{L}_{i,j} \text{ with } \mathcal{C}(x_k) = i \text{ and} \\ a_{i,m}(x_l) = a_{j,n}(x_l) \text{ for } x_l \in \mathcal{L}_{i,j} \end{array} \right\}$$

Capacities etc. are introduced in the respective reductions. □

Definition 15 (Request Graph $r(\phi)$)**.** 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. $V_{r(\phi)} = \{v_i \,|\, \mathcal{C}_i \in \mathcal{C}_\phi\}$. Matching the construction of the substrate graph $G_{S(\phi)}$, we introduce directed edges $(v_i, v_j) \in E_{r(\phi)}$ only if there exists a literal $x_k \in \mathcal{C}_i$ being introduced in $\mathcal{C}_i$ and being also used in the clause $\mathcal{C}_j$:

$$E_{r(\phi)} = \{(v_i, v_j) \,|\, \exists x_k \in \mathcal{L}_{i,j} \text{ with } \mathcal{C}(x_k) = i\}$$

Demands etc. are introduced in the respective reductions. □

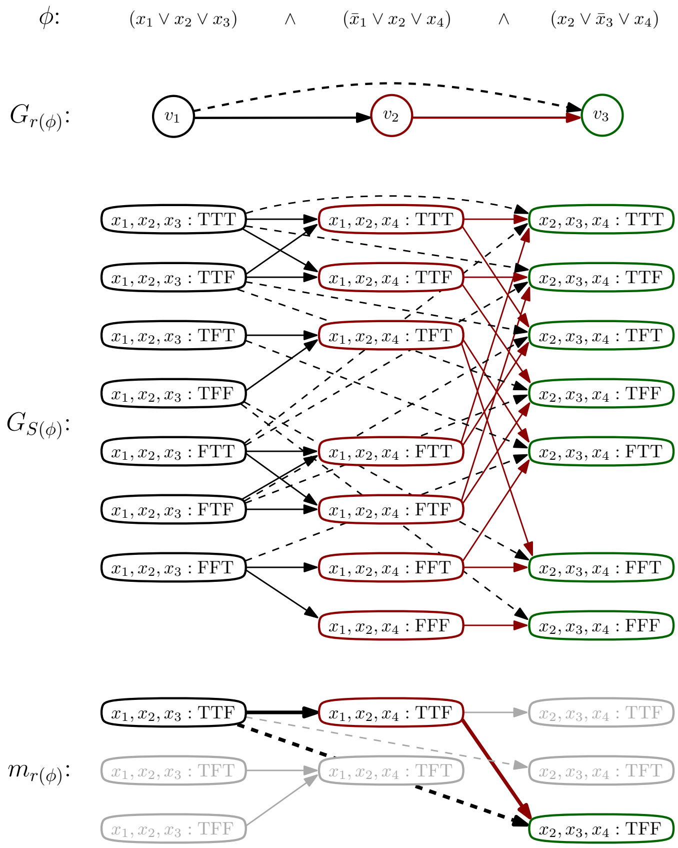

Fig. 2. Visualization of the construction of substrate and request graphs for the 3-SAT formula ϕ (cf. Definitions 14 and 15). Additionally, a mapping m satisfying the conditions of Lemma 16 is shown. Accordingly, the formula ϕ is satisfied. Concretely, the mapping represents the assignment of truth values $x_1 = \mathrm{T}$, $x_2 = \mathrm{T}$, $x_3 = \mathrm{F}$, $x_4 = \mathrm{F}$.

C. The Base Lemma

Nearly all of our results are based on the following lemma.

Lemma 16. *The 3-*SAT *formula ϕ is satisfiable*

if and only if

there exists a valid mapping m of $G_{r(\phi)}$ on $G_{S(\phi)}$, such that

1) *each virtual node v_i is mapped on a substrate node corresponding to assignments $\mathcal{A}_i$ of the i-th clause, i.e. $m_V(v_i) \in \mathcal{A}_i$ holds for all $v_i \in V_{r(\phi)}$, and*
2) *virtual edges are embedded using a single substrate edge, i.e. $|m_E(v_i, v_j)| = 1$ holds for all $(v_i, v_j) \in E_{r(\phi)}$.*

Proof. We first show that if ϕ is satisfiable, then such a mapping m must exist. Afterwards, we show that if such a mapping m 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 $m_V(v_j)$. This edge $(m_V(v_i), m_V(v_j))$ must exist in $E_{S(\phi)}$, as the existence of virtual edge (v_i, v_j) implies that clause $\mathcal{C}_i$ is the first clause introducing a literal of $\mathcal{L}_{i,j}$ and $m_V(v_i) = a_{i,m}$ and $m_V(v_j) = a_{j,n}$ must agree by construction on the assignment of truth values for all literals. Clearly, the constructed mapping m fulfills both the conditions stated in the lemma, hence completing the first half of the proof.

We now show that if there exists a mapping m meeting the two requirements stated in the lemma, 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 clauses $\mathcal{C}_\phi$ one by one (according to the precedence relation of the indices). 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. $\mathcal{L}_i \cap \bigcup_{j<i} \mathcal{L}_j = \emptyset$ 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 $m_V(v_j) = a_{j,n} \in \mathcal{A}_j$. As the edge (v_j, v_i) is contained in $E_{r(\phi)}$ by definition and all edges are mapped using a single substrate edge by our assumptions, $m_E(v_i, v_j) = \langle (a_{j,n}, a_{i,m}) \rangle$ must hold. Hence, as $(a_{j,n}, a_{i,m}) \in E_{S(\phi)}$ must hold and edges are only introduced *if* assignments agree with each other, we have $\big[m_V(v_j)\big](x_k) = a_{j,n}(x_k) = a_{i,m}(x_k) = \big[m_V(v_i)\big](x_k)$. 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, α satisfies all of the clauses and hence satisfies ϕ, completing the proof of the base lemma. ∎

The base lemma is the heart of our reduction framework for obtaining our results and we note that the construction of the substrate and the request graph is polynomial in the size of the 3-SAT formula. Indeed, the base lemma forms the basis for polynomial-time reductions for the different VNEP decision variants. Concretely, consider some VNEP variant $\langle\, \mathbf{X} \,|\, \mathbf{Y} \,\rangle$. If this variant is 'expressive' enough such that any feasible embedding must meet the criteria of Lemma 16, then $\langle\, \mathbf{X} \,|\, \mathbf{Y} \,\rangle$ is – by reduction from 3-SAT – $\mathcal{NP}$-hard. Furthermore, the existence of an Integer Program for each of the VNEP variants (cf. technical report [22]) shows that the variants lie in $\mathcal{NP}$ and hence the successful application of the base lemma shows the $\mathcal{NP}$-completeness of the respective variants. As a result, for the considered VNEP variants, any optimization problem (e.g. cost) cannot be approximated within any factor. The following lemma formalizes this observation:

Lemma 17. *If there is a polynomial-time reduction from 3-*SAT *to the* VNEP *decision variant* $\langle\, \mathbf{X} \,|\, \mathbf{Y} \,\rangle$*, then the* VNEP *variant* $\langle\, \mathbf{X} \,|\, \mathbf{Y} \,\rangle$ *is* $\mathcal{NP}$*-complete. Furthermore, any optimization problem over the same set of constraints is (i)* $\mathcal{NP}$*-hard and (ii) inapproximable (within any factor), unless* $\mathcal{P} = \mathcal{NP}$ *holds.*

Lastly, the following lemma will prove useful when applying the base lemma.

Lemma 18. *Exactly one of the following two following properties holds for formula* φ*:*

1) *The clauses of* ϕ *can be ordered such that within the corresponding request graph* $G_{r(\phi)}$ *only the node* $v_1 \in V_{r(\phi)}$ *has no incoming edges.*
2) φ *can be decomposed into formulas* φ_1 *and* φ_2*, such that the sets of literals occurring in* φ_1 *and* φ_2 *are disjoint, while* $\varphi = \varphi_1 \wedge \varphi_2$ *holds. Hence,* φ *is satisfiable iff.* φ_1 *and* φ_2 *are (independently) satisfiable.*

Proof. We prove the statement by a greedy construction and assume that the clauses are initially unordered. We iteratively assign an index to the clauses, keeping track of which clauses were not assigned an index yet. Initially, pick any of the clauses and assign it the index 1. Now, iteratively choose any clause which contains a literal that already occurs in the set of indexed clauses. If no such clause exists, then the clauses already indexed and the clauses not indexed obviously represent a partition of the literal set and hence the second statement holds true. However, if the greedy step succeeded every time, then the following holds with respect to the constructed ordering: any virtual node v_i corresponding to clause $\mathcal{C}_i$, for $i > 1$, must have an incoming edge by Definition 15 as the clause overlapped with the already introduced literals. ∎

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

We employ our reduction framework outlined in the previous section to derive a series of hardness results for the VNEP. In particular, we first show the $\mathcal{NP}$-completeness of the original VNEP variant $\langle\, \mathbf{VE} \,|\, \textbf{-} \,\rangle$ in the absence of additional restrictions. Given this result, we investigate several other problem settings and show, among others, that also deciding $\langle\, \textbf{-} \,|\, \mathbf{LN} \,\rangle$ is $\mathcal{NP}$-complete. Hence, even when the physical network does not impose any resource constraints (i.e., nodes and edges have infinite capacities), finding an embedding satisfying latency and node placement restrictions is $\mathcal{NP}$-complete. Again, it must be noted that adding further restrictions only renders the VNEP harder (cf. Lemma 9).

A. $\mathcal{NP}$-Completeness under Capacity Constraints

We first consider the most basic VNEP variant $\langle\, \mathbf{VE} \,|\, \textbf{-} \,\rangle$.

Theorem 19. VNEP $\langle\, \mathbf{VE} \,|\, \textbf{-} \,\rangle$ *is* $\mathcal{NP}$*-complete and cannot be approximated under any objective (unless* $\mathcal{P} = \mathcal{NP}$*).*

Proof. We show the statement via a polynomial-time reduction from 3-SAT according to Lemmas 16 and 17. Given is a 3-SAT formula ϕ. We assume for now that the first statement of Lemma 18 holds, i.e. that within the request graph $G_{r(\phi)}$ only the first node $v_1 \in V_{r(\phi)}$ has no incoming edge.

To enforce the properties of Lemma 16, we set the substrate and request capacities for some small λ, $0 < \lambda < 1/|\mathcal{C}_\phi|$, as follows. The capacity of substrate nodes is determined by the clause whose assignments they represent. Furthermore, the capacities decrease monotonically with each clause. Similarly, but now increasing per clause, the capacities of edges are determined by the clause that the edge's head corresponds to:

$$\begin{aligned} c_S(a_{i,m}) &= 1 + \lambda \cdot (M - i) && \forall\, \mathcal{C}_i \in \mathcal{C}_\phi, a_{i,m} \in \mathcal{A}_i \\ c_S(e) &= 1 + \lambda \cdot i && \forall\, \mathcal{C}_i \in \mathcal{C}_\phi, e \in \delta^-(\mathcal{A}_i) \end{aligned}$$

The demands are set to match the respective capacities:

$$\begin{aligned} c_{r(\phi)}(v_i) &= 1 + \lambda \cdot (M - i) && \forall\, v_i \in V_{r(\phi)} \\ c_{r(\phi)}(e) &= 1 + \lambda \cdot i && \forall\, v_j \in V_{r(\phi)}, e \in \delta^-(v_j) \end{aligned}$$

Due to the decreasing node demands and capacities, virtual node $v_j \in V_{r(\phi)}$ corresponding to clause $\mathcal{C}_i$ can only be mapped on substrate nodes $\bigcup_{k=1}^{j} \mathcal{A}_k$. Due to the choice of λ, the capacity of any substrate node is less than 2 while each virtual node has a demand larger than 1. Hence, two virtual nodes can never be collocated (mapped) on the same substrate node. Thus, all virtual edges must be mapped onto at least a single substrate edge. Considering the virtual edge $e = (v_i, v_j) \in E_{r(\phi)}$ with demand $c_{r(\phi)}(e) = 1 + \lambda \cdot j$, the virtual node v_j must be mapped on a substrate node having an incoming edge of at least capacity $1 + \lambda \cdot j$. As the edge capacities increase with the clause index, only the substrate nodes in $\bigcup_{k=j}^{M} \mathcal{A}_k$ satisfy this condition. Hence, if node v_j has an incoming edge, it can only be mapped on nodes in $\bigcup_{k=1}^{j} \mathcal{A}_k \cap \bigcup_{k=j}^{M} \mathcal{A}_k = \mathcal{A}_j$. As we assumed that the first statement of Lemma 18 holds for ϕ and hence all nodes $v_2, \ldots, v_M$ have an incoming edge, we obtain that the virtual node v_i must be mapped on $\mathcal{A}_i \subseteq V_{S(\phi)}$ for $i = 2, \ldots, M$. Considering the first node v_1, we observe that only nodes in $\mathcal{A}_1$ offer sufficient capacity to host v_1. Hence, any feasible embedding will obey the first statement of Lemma 16 regarding the node mappings.

We now show that *any* feasible mapping will also obey the second property of Lemma 16, namely, that any virtual edge is mapped on exactly one substrate edge. To this end, assume for the sake of contradiction that $(v_i, v_j) \in E_{r(\phi)}$ 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 via 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 $a_{k,l}$. Hence, the only feasible edges for embedding the respective virtual edges are the direct connections between any two substrate nodes.

Therefore, all *feasible* solutions indicate the satisfiability of the formula ϕ. Any algorithm computing a feasible solution to the VNEP obeying node and edge capacities, decides 3-SAT.

Lastly, we argue for the validity of our assumption on the structure of ϕ, namely that the first statement of Lemma 18 holds. If this were not to hold, then the second statement of Lemma 18 holds true and the formula can be decomposed (potentially multiple times) into disjoint subformulas $\varphi_1, \ldots, \varphi_k$, such that (i) $\varphi = \bigwedge_{i=1}^{k} \varphi_i$ holds, and (ii) such that the first condition of Lemma 18 holds for each subformula. Accordingly, assuming that an algorithm exists which can construct feasible embeddings whenever they exist, this algorithm can be used to decide the satisfiability of each subformula, hence deciding the original satisfiability problem. ■

B. $\mathcal{NP}$-completeness under Additional Constraints

Building on the above $\mathcal{NP}$-completeness proof, we can adapt it easily to other settings.

Theorem 20. VNEP $\langle\, \mathbf{E} \,|\, \mathbf{N} \,\rangle$ *is $\mathcal{NP}$-complete and cannot be approximated under any objective (unless $\mathcal{P} = \mathcal{NP}$).*

Proof. In this setting node placement restrictions and substrate edge capacities are enforced. We apply the same construction as in the proof of Theorem 19. 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 $v_i \in V_{r(\phi)}$. By the same argument as before, virtual edges cannot be mapped onto paths as the intermediate nodes do not support the respective demand. ■

Theorem 21. VNEP $\langle\, \mathbf{V} \,|\, \mathbf{R} \,\rangle$ *is $\mathcal{NP}$-complete and cannot be approximated under any objective (unless $\mathcal{P} = \mathcal{NP}$).*

Proof. In this setting only node capacities must be obeyed, while routing restrictions may be introduced. We employ the same node capacities as in the proof of Theorem 19, such that virtual node $v_i \in V_{r(\phi)}$ may only be mapped on nodes $\bigcup_{k=1}^{i} \mathcal{A}_k$. Routing restrictions are set to only allow direct edges, i.e. $\overline{E}_S^{v_i,v_j} = E_{S(\phi)} \setminus (\mathcal{A}_i \times \mathcal{A}_j)$ holds for each $(v_i, v_j) \in E_{r(\phi)}$. Again, $v_1 \in V_{r(\phi)}$ must be mapped on a node in $\mathcal{A}_1$, while all other virtual nodes have at least one incoming edge according to Lemma 18. As multiple virtual nodes cannot be placed on the same substrate node and virtual edges must span at least one substrate edge, any node v_j can only be mapped on nodes in $\mathcal{A}_j$ for $j \in \{2, \ldots, M\}$. Together with the routing restrictions both requirements of Lemma 16 are safeguarded and the result follows. ■

Theorem 22. VNEP *variants* $\langle\, \textbf{-} \,|\, \mathbf{NR} \,\rangle$ *and* $\langle\, \textbf{-} \,|\, \mathbf{NL} \,\rangle$ *are $\mathcal{NP}$-complete and cannot be approximated under any objective (unless $\mathcal{P} = \mathcal{NP}$).*

Proof. In both cases capacities are not considered at all. Allowing for node placement restrictions, the first property of Lemma 16 is easily safeguarded (cf. proof of Theorem 20). By employing the same routing restrictions as in the proof of Theorem 21 the result follows directly for the case $\langle\, \textbf{-} \,|\, \mathbf{NR} \,\rangle$. Latency restrictions can be easily used to enforce that virtual edges do not span more than a single substrate edge. Concretely, we set unit substrate edge latencies and unit virtual edge latency bounds: if an edge was to be embedded via more than one edge, the latency restrictions would be violated. Hence, the result also holds for $\langle\, \textbf{-} \,|\, \mathbf{NL} \,\rangle$. ■

V. $\mathcal{NP}$-COMPLETENESS OF COMPUTING APPROXIMATE EMBEDDINGS

Given the hardness results presented in Section IV, the question arises to which extent the hardness can be overcome when only computing approximate embeddings (cf. Definition 10), i.e. embeddings that may violate capacity constraints or exceed latency constraints by certain factors. Based on the proofs presented in Section IV, we first derive hardness results for computing α-approximate embeddings (allowing node capacity violations) and γ-approximate embeddings (allowing latency violations). For β-approximate embeddings (allowing edge capacity violations) a reduction from an edge-disjoint paths problem is presented in our technical report [22].

Theorem 23. *For* $\langle\, \mathbf{VE} \,|\, \textbf{-} \,\rangle$ *and* $\langle\, \mathbf{V} \,|\, \mathbf{R} \,\rangle$ *finding an α-approximate embedding is $\mathcal{NP}$-complete as well as inapproximable under any objective (unless $\mathcal{P} = \mathcal{NP}$) for any $\alpha < 2$.*

Proof. Assume that there exists an algorithm computing α-approximate embeddings for $\alpha = 2 - \varepsilon$, $0 < \varepsilon < 1$. We adapt the proofs of Theorem 19 and 21 slightly. First, note that for α-approximate mappings validity still has to hold according to Definition 10. Hence, by the decreasing node capacities the virtual node v_j can only be mapped on substrate nodes $\bigcup_{k=1}^{j} \mathcal{A}_k$. Furthermore, by either enforcing edge capacities or edge routing restrictions, the node v_j can still only be mapped on $\mathcal{A}_j$. Hence, the only missing piece to show that the respective proofs still hold is the fact that still at most a single virtual node can be mapped on a single substrate node. To ensure, that this still holds, we adapt the capacities. Concretely, we choose λ, such that $\lambda < \varepsilon / (2 \cdot |\mathcal{C}_\phi|)$ holds. Hence, the capacity of any substrate node is less than $1 + \varepsilon/2$. By relaxing the capacity constraints by the factor $2 - \varepsilon$, the allowed substrate node allocations are upper bounded by $(1 + \varepsilon/2) \cdot (2 - \varepsilon) = 2 - \varepsilon - \varepsilon^2/2 < 2$. As the demand of any virtual node is larger than 1, still at most a single virtual node can be mapped on a substrate node. Hence, the respective proofs still apply and the results follow. ■

The result for γ-approximate embeddings can be obtained in a very similar fashion.

Theorem 24. *For $\langle\,\textbf{-}\,|\,\textbf{NL}\,\rangle$ finding an γ-approximate embedding is $\mathcal{NP}$-complete as well as inapproximable under any objective (unless $\mathcal{P}=\mathcal{NP}$) for any $\gamma < 2$.*

Proof. The proof of Theorem 22 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. Hence, the result of Theorem 22 remains valid. ∎

For β-approximate embeddings similar results can be obtained via a reduction from an edge-disjoint paths problem. Due to space constraints we only state our main result and refer the reader to our technical report for the proof [22].

Theorem 25. *Finding a β-approximate embedding for the* VNEP *variants $\langle\,\textbf{VE}\,|\,\textbf{-}\,\rangle$ and $\langle\,\textbf{E}\,|\,\textbf{N}\,\rangle$ is hard to approximate for $\beta \in \Theta(\log n/\log\log n)$, $n = |V_S|$, unless $\mathcal{NP} \subseteq \mathcal{BP}$-TIME$(\bigcup_{d\geq 1} n^{d\log\log n})$ holds.*

Note that above $\mathcal{BP}$-*TIME*$(f(n))$ denotes the class of problems solvable by probabilistic Turing machines in time $f(n)$ with bounded error-probability [24].

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

All of our $\mathcal{NP}$-completeness results (except Theorem 25) are based on a reduction from 3-SAT, yielding a specific directed-acylic substrate graph $G_{S(\phi)}$ and a specific directed acyclic request graph $G_{r(\phi)}$ and we note the following.

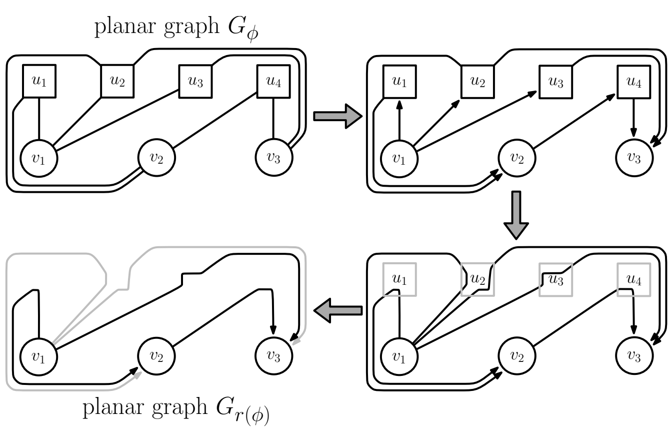

Fig. 3. Depicted is the transformation process of a planar graph G_ϕ (cf. Definition 27) to the planar graph $G_{r(\phi)}$. Concretely, the example formula of Figure 2 is revisited, i.e. $\phi = C_1 \wedge C_2 \wedge C_3$, with $C_1 = x_1 \vee x_2 \vee x_3$, $C_2 = \bar{x}_1 \vee x_2 \vee x_4$, and $C_3 = x_2 \vee \bar{x}_3 \vee x_4$. In the first step all edges are directed, such that edges from clause 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). 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.

Observation 26. Theorems 19 - 24 still hold when restricting the request and the substrate to acyclic graphs.

Given the hardness of the VNEP and as for example Virtual Clusters (an undirected star network) can be optimally embedded in polynomial time [7], one might ask whether the hardness is preserved when restricting request graphs further.

In this section, we derive the result that the VNEP variants considered in this paper remain $\mathcal{NP}$-complete when request graphs are planar and degree-bounded. Our results are obtained by considering a *planar* variant of 3-SAT. The planarity of a formula ϕ is defined according ϕ's graph interpretation:

Definition 27 (Graph G_ϕ of formula ϕ)**.** The graph $G_\phi = (V_\phi, E_\phi)$ of a SAT formula ϕ is defined as follows. V_ϕ contains a node v_i for each clause $C_i \in \mathcal{C}_\phi$ and a node u_k for each literal $x_k \in \mathcal{L}_\phi$. An undirected edge $\{v_i, u_k\}$ is contained in E_ϕ, iff. the literal x_k is contained in C_i (either positive or negative). Note that the graph G_ϕ is bipartite. □

An example for the interpretation G_ϕ is depicted in Figure 3. Kratochvíl [25] considered the following variant of 4P3C-3-SAT and proved its $\mathcal{NP}$-completeness.

Definition 28 (4P3C-3-SAT)**.** The 4-Bounded Planar 3-Connected 3-SAT (4P3C-3-SAT) considers only 3-SAT formulas ϕ for which the following holds:
(1) In each clause, exactly 3 distinct literals are used.
(2) Each literal occurs in at most 4 clauses.
(3) The graph G_ϕ is planar.
(4) The graph G_ϕ is vertex 3-connected. □

Theorem 29 ([25])**.** 4P3C-3-SAT *is $\mathcal{NP}$-complete.*

The following lemma connects 4P3C-3-SAT formulas ϕ with the corresponding request graphs $G_{r(\phi)}$.

Lemma 30. *Given a* 4P3C-3-SAT *formula ϕ, the following holds for the request graph $G_{r(\phi)}$ (cf. Definition 15):*
1) The request graph $G_{r(\phi)}$ is planar.
2) The node-degree of $G_{r(\phi)}$ is bounded by 12.

Proof. We consider an arbitrary 4P3C-3-SAT formula ϕ to which the conditions of Definition 28 apply. We first show that the corresponding request graph $G_{r(\phi)}$ is planar by detailing a transformation process leading from G_ϕ to $G_{r(\phi)}$ while preserving planarity (see Figure 3 for an illustration).

Starting with the undirected graph G_ϕ, the edges are first oriented: an edge 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. Note that while many reductions in Section IV required the reordering of clause nodes according to Lemma 18, this reordering preserves planarity as the structure of the graph G_ϕ does not change.

Given this directed graph, the literal nodes are now removed by joining the single incoming edge of the literal nodes with *each* outgoing edge of the corresponding literal node. In particular, consider the literal node x_2 of Figure 3: the single incoming edge (C_1, x_2) is joined with the outgoing edges (x_2, C_2) and (x_2, C_3) to obtain the edges (C_1, C_2)

and (C_1, C_3), respectively. 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 12 (in the undirected interpretation of the graph $G_{r(\phi)}$). To see this, we note the following: based on the first two conditions of Definition 28, each clause node connects to exactly 3 literal nodes and each literal node connects to at most 4 clause nodes. Hence, when removing the literal nodes in the transformation process, the degree of each node may increase at most by a factor of 4. As any clause node had 3 neighboring literal nodes, this implies that the degree of any node is at most 12 after the transformation process, completing the proof. ■

Given the above, we easily derive the following theorem:

Theorem 31. *Theorems 19 - 24 hold when restricting the request graphs to be planar and / or degree 12-bounded. Theorem 25 holds for planar and degree 1-bounded graphs.*

Proof. Our $\mathcal{NP}$-completeness proofs in Section IV and Section V (except for Theorem 25) relied solely on the reduction from 3-SAT to VNEP using the base Lemma 16. As formulas of 4P3C-3-SAT are a strict subset of the 3-SAT formulas, the base Lemma 16 is still applicable for 4P3C-3-SAT formulas. However, due to the structure of 4P3C-3-SAT formulas, the corresponding requests in the reductions are planar and exhibit a node-degree bound of 12 by Lemma 30. Hence, solving the VNEP is $\mathcal{NP}$-complete, even when restricting the requests to planar and / or degree-bounded ones. Lastly, as proven in our technical report [22], Theorem 25 holds for planar and degree 1-bounded requests, concluding the proof. ■

VII. Conclusion

We presented a comprehensive set of hardness results for the VNEP and its variants, which lie at the core of many resource allocation problems in networks. Our results are negative in nature: we show that the problem variants are $\mathcal{NP}$-complete and hence inapproximable (unless $\mathcal{P} = \mathcal{NP}$) and that this holds true even for restricted classes of request graphs.

We believe that our results are of great importance for future work on several of the virtual network embedding problems. For example, our results on the variant enforcing node placement and latency restrictions are of specific interest for Service Function Chaining. Surprisingly, the respective problem is hard even when not considering any capacity constraints. Furthermore, we have shown that it is hard to compute embeddings satisfying latency bounds within a factor of (less than) two times the original bounds. In turn, whenever latency bounds must be obeyed strictly, one needs to rely on exact algorithmic techniques as e.g. Integer Programming.

Acknowledgements

This work was partially supported by Aalborg University's PreLytics project as well as by the German BMBF Software Campus grant 01IS1205.

References

[1] J. C. Mogul and L. Popa, "What we talk about when we talk about cloud network performance," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 5, 2012.

[2] N. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *IEEE INFOCOM*, 2009.

[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, 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)*, 2014.

[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*, New York, NY, USA, 2011.

[7] M. Rost, C. Fuerst, and S. Schmid, "Beyond the stars: Revisiting virtual cluster embeddings," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 3, 2015.

[8] G. Even, M. Medina, G. Schaffrath, and S. Schmid, "Competitive and deterministic embeddings of virtual networks," *Theoretical Computer Science*, vol. 496.

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

[10] M. Chowdhury, M. R. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, Feb. 2012.

[11] G. Schaffrath, S. Schmid, and A. Feldmann, "Optimizing long-lived cloudnets with migrations," in *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*, 2012.

[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 *Proceedings of the 8th International Conference on Network and Service Management*. IFIP, 2013.

[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. Springer, 2011.

[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. Springer, 2016.

[15] M. Rost and S. Schmid, "Virtual Network Embedding Approximations: Leveraging Randomized Rounding," in *(to appear) Proceedings IFIP Networking 2018*, 2018, preprint available at arXiv:1803.03622 [cs.NI]. [Online]. Available: http://arxiv.org/abs/1803.03622.

[16] ——, "(FPT-)Approximation Algorithms for the Virtual Network Embedding Problem," Tech. Rep. arXiv:1803.04452 [cs.NI], March 2018. [Online]. Available: http://arxiv.org/abs/1803.04452

[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*, 2007.

[18] J. Díaz, J. Petit, and M. Serna, "A survey of graph layout problems," *ACM Computing Surveys*, vol. 34, no. 3, Sep. 2002.

[19] D. Eppstein, "Subgraph isomorphism in planar graphs and related problems," in *Graph Algorithms and Applications I*, 2011.

[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] M. Rost and S. Schmid, "NP-Completeness and Inapproximability of the Virtual Network Embedding and Its Variants," *CoRR*, vol. abs/1801.03162, 2018. [Online]. Available: http://arxiv.org/abs/1801.03162

[23] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*. Springer, 1972, pp. 85–103.

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

[25] J. Kratochvíl, "A special planar satisfiability problem and a consequence of its np-completeness," *Discrete Applied Mathematics*, vol. 52, no. 3, 1994.