Virtual Network Embedding Approximations: Leveraging Randomized Rounding

Matthias Rost
TU Berlin, Germany
Email: mrost@inet.tu-berlin.de

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

Abstract—**The Virtual Network Embedding Problem (VNEP) captures the essence of many resource allocation problems of today's infrastructure providers, which offer their physical computation and networking resources to customers. Customers request resources in the form of Virtual Networks, i.e. as a directed graph which specifies computational requirements at the nodes and communication requirements on the edges. An embedding of a Virtual Network on the shared physical infrastructure is the joint mapping of (virtual) nodes to physical servers together with the mapping of (virtual) edges onto paths in the physical network connecting the respective servers.**

This work initiates the study of approximation algorithms for the VNEP. Concretely, we study the offline setting with admission control: given multiple request graphs the task is to embed the most profitable subset while not exceeding resource capacities. Our approximation is based on the randomized rounding of Linear Programming (LP) solutions. Interestingly, we uncover that the standard LP formulation exhibits an inherent structural deficit when considering general virtual networks: its solutions cannot be decomposed into valid embeddings. In turn, focusing on the class of cactus request graphs, we devise a novel LP formulation, whose solutions can be decomposed into convex combinations of valid embedding. Proving performance guarantees of our rounding scheme, we obtain the first approximation algorithm for the VNEP in the resource augmentation model.

We propose two rounding heuristics and evaluate their performance in an extensive computational study, showing that these consistently yield good solutions (even without augmentations).

I. Introduction

Cloud applications usually consist of multiple distributed components (e.g., virtual machines, containers), which results in substantial communication requirements. If the provider fails to ensure that these communication requirements are met, the performance can suffer dramatically [1]. Consequently, over the last years, several proposals have been made to *jointly* provision the computational functionality *together* with appropriate network resources. The Virtual Network Embedding Problem (VNEP) captures the core of this problem: given a directed graph specifying computational requirements at the nodes and bandwidth requirements on the edges, an embedding of this *Virtual Network* in the physical network has to be found, such that both the computational and the network requirements are met. Figure 1 illustrates two incarnations of virtual networks: *service chains* [2] and *virtual clusters* [3].

We study the offline setting with admission control: given multiple requests the task is to embed the most profitable subset while not exceeding resource capacities.

ISBN 978-3-903176-08-9 © 2018 IFIP

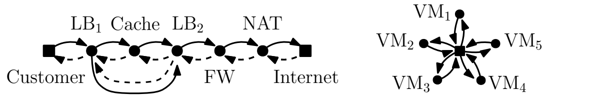

Fig. 1. Examples for virtual networks 'in the wild'. The left graph shows a service chain for mobile operators [4]: load-balancers route (parts of the) traffic through a cache. Furthermore, a firewall and a network-address translation are used. The right graph depicts the Virtual Cluster abstraction for provisioning virtual machines (VMs) in data centers. The abstraction provides connectivity guarantees via a *logical switch* in the center [3].

A. Formal Problem Statement

In the light of the recent interest in Service Chaining [2], we extend the VNEP's general definition [5] by considering different *types* of computational nodes. We refer to the physical network as the *substrate network*. The substrate $G_S = (V_S, E_S)$ is offering a set $\mathcal{T}$ of computational types. This set of types may contain e.g., 'FW' (firewall), 'x86 server', etc. For a type $\tau \in \mathcal{T}$, the set $V_S^\tau \subseteq V_S$ denotes the substrate nodes that can host functionality of type τ. Denoting the node resources by $R_S^V = \{(\tau, u) \mid \tau \in \mathcal{T}, u \in V_S^\tau\}$ and all substrate resources by $R_S = R_S^V \cup E_S$, the capacity of nodes and edges is denoted by $d_S(x,y) > 0$ for $(x,y) \in R_S$.

For each request $r \in \mathcal{R}$, a directed graph $G_r = (V_r, E_r)$ together with a profit b_r is given. We refer to the respective nodes as virtual or request nodes and similarly refer to the respective edges as virtual or request edges. The types of virtual nodes are indicated by the function $\tau_r : V_r \to \mathcal{T}$.

Based on policies of the customer or the provider, the mapping of virtual node $i \in V_r$ is restricted to a set $V_S^{r,i} \subseteq V_S^{\tau_r(i)}$, while the mapping of virtual edge (i,j) is restricted to $E_S^{r,i,j} \subseteq E_S$. Each virtual node $i \in V_r$ and each edge $(i,j) \in E_r$ is attributed with a resource demand $d_r(i) \geq 0$ and $d_r(i,j) \geq 0$, respectively. Virtual nodes and edges can only be mapped on substrate nodes and edges of sufficient capacity, i.e. $V_S^{r,i} \subseteq \{u \in V_S^{\tau_r(i)} | d_S(u) \geq d_r(i)\}$ and $E_S^{r,i,j} \subseteq \{(u,v) \in E_S | d_S(u,v) \geq d_r(i,j)\}$ holds.

We denote by $d_{\max}(r,x,y)$ the maximal demand that a request r may impose on a resource $(x,y) \in R_S$:

$$d_{\max}(r,\tau,u) = \max(\{0\} \cup \{d_r(i) | i \in V_r : \tau(i) = \tau \wedge u \in V_S^{r,i}\})$$
$$d_{\max}(r,u,v) = \max(\{0\} \cup \{d_r(i,j) | (i,j) \in E_r : (u,v) \in E_S^{r,i,j}\})$$

In the following the notions of valid mappings (respecting mapping constraints) and feasible embeddings (respecting resource constraints) are introduced to formalize the VNEP.

Definition 1 (Valid Mapping)**.** *A valid mapping m_r of request $r \in \mathcal{R}$ is a tuple (m_r^V, m_r^E) of functions $m_r^V : V_r \to V_S$ and $m_r^E : E_r \to \mathcal{P}(E_S)$, such that the following holds:*

- *Virtual nodes are mapped to allowed substrate nodes: $m_r^V(i) \in V_S^{r,i}$ holds for all $i \in V_r$.*
- *The mapping $m_r^E(i,j)$ of virtual edge $(i,j) \in E_r$ is an edge-path connecting $m_r^V(i)$ to $m_r^V(j)$ only using allowed edges, i.e. $m_r^E(i,j) \subseteq \mathcal{P}(E_S^{r,i,j})$ holds.*

We denote by $\mathcal{M}_r$ the set of valid mappings of request $r \in \mathcal{R}$.

Definition 2 (Allocations of Valid Mappings)**.** *We denote by $A(m_r, x, y)$ the cumulative allocation induced by the valid mapping $m_r \in \mathcal{M}_r$ on resource $(x,y) \in R_S$:*

$$A(m_r, \tau, u) = \sum\nolimits_{i \in V_r, \tau(i)=\tau, m_r^V(i)=u} d_r(i) \qquad \forall (\tau, u) \in R_S^V$$

$$A(m_r, u, v) = \sum\nolimits_{(i,j) \in E_r, (u,v) \in m_r^E(i,j)} d_r(i,j) \ \forall (u,v) \in E_S$$

The maximal allocation that a valid mapping of request $r \in \mathcal{R}$ may impose on a substrate resource $(x,y) \in R_S$ is denoted by $A_{\max}(r,x,y) = \max_{m_r \in \mathcal{M}_r} A(m_r, x, y)$.

Definition 3 (Feasible Embedding)**.** *A feasible embedding of a subset of requests $\mathcal{R}' \subseteq \mathcal{R}$ is a collection of valid mappings $\{m_r\}_{r \in \mathcal{R}'}$, such that the* cumulative *allocations on nodes and edges does not exceed the substrate capacities, i.e. $\sum_{r \in \mathcal{R}'} A(m_r, x, y) \leq d_S(x,y)$ holds for $(x,y) \in R_S$.*

Definition 4 (Virtual Network Embedding Problem)**.** *The VNEP asks for a feasible embedding $\{m_r\}_{r \in \mathcal{R}'}$ of a subset of requests $\mathcal{R}' \subseteq \mathcal{R}$ maximizing the profit $\sum_{r \in \mathcal{R}'} b_r$.*

B. Related Work

In the last decade, the VNEP has attracted much attention due to its many applications and the survey [5] from 2013 already lists more than 80 different algorithms for its many variations [5]. The VNEP is known to be $\mathcal{NP}$-hard and inapproximable in general (unless $\mathcal{P} = \mathcal{NP}$) [6]. Based on the hardness of the VNEP, most works consider heuristics without any performance guarantee [5], [7]. Other works proposed exact methods as integer or constraint programming, coming at the cost of an exponential runtime [8], [9], [10].

A column generation approach was proposed by Jarray et al. in [9] to efficiently compute solutions to the VNEP by generating valid mappings 'on-the-fly'. We believe that our decomposable LP formulations may be used to price (i.e. generate) further valid mappings more efficiently than by using Mixed-Integer Programming.

Acknowledging the hardness of the general VNEP and the diversity of applications, several subproblems of the VNEP have been studied recently by considering restricted graph classes for the virtual networks and the substrate graph. For example, virtual clusters with uniform demands are studied in [11], [3], line requests are studied in [12], [13], [14] and tree requests were studied in [15], [13].

Considering approximation algorithms, Even et al. employed randomized rounding in [13] to obtain a constant approximation for embedding line requests on arbitrary substrate graphs under strong assumptions on *both* the benefits and the capacities. In their interesting work, Bansal et al. [15] give an $n^{O(d)}$ time $O(d^2 \log(nd))$-approximation algorithm for minimizing the load of embedding d-depth trees based on a strong LP relaxation inspired by the Sherali-Adams hierarchy. To the best of our knowledge, no approximation algorithms are known for arbitrary substrate graphs and classes of virtual networks containing cyclic substructures.

Bibliographic Note: In our preliminary technical report [16] similar results were presented. The current work presents a significantly simpler LP formulation and also provides an extensive computational evaluation. An extended version of this work, containing all proofs and additional details on our evaluation, can be found at [17].

Additionally, in our recent technical report [18], the approximation approach presented in this work is extended beyond cactus request graphs. However, approximating more general request graphs comes at the price of non-polynomial runtimes.

C. Outline of Randomized Rounding for the VNEP

We shortly revisit the concept of randomized rounding. Given an Integer Program for a certain problem, randomized rounding works by (i) computing a solution to its Linear Program relaxation, (ii) decomposing this solution into convex combinations of *elementary* solutions, and (iii) probabilistically selecting elementary solutions based on their weight.

Accordingly, for applying randomized rounding for the VNEP, a convex combination of valid mappings $\mathcal{D}_r = \{(f_r^k, m_r^k) | m_r^k \in \mathcal{M}_r, f_r^k > 0\}$ must be recovered from the Linear Programming solution for each request $r \in \mathcal{R}$, such that (i) the profit of these convex combinations equals the profit achieved by the Linear Program and (ii) the (fractional) cumulative allocations do not violate substrate capacities. To round a solution, for each request r the mapping m_r^k is selected with probability f_r^k, rejecting r with probability $1 - \sum_k f_r^k$.

D. Results and Organization

This paper initiates the study of approximation algorithms for the VNEP on general substrates *and* general virtual networks. Specifically, we employ randomized rounding to obtain the first approximation algorithm for the non-trivial class of cactus graph requests in the resource augmentation model.

Studying the classic multi-commodity flow (MCF) formulation for the VNEP in Section II, we show that its solutions can only be decomposed for tree requests: request graphs containing cycles can in general not be decomposed into valid mappings. This result has ramifications beyond the inability to apply randomized rounding: we prove that the MCF formulation exhibits an unbounded integrality gap. Investigating the root cause for this surprising result, we devise a novel *decomposable* Linear Programming formulation in Section III for the class of cactus graph requests. We then present and prove performance guarantees for our randomized rounding algorithm in Section IV, obtaining the first approximation algorithm for the Virtual Network Embedding Problem. Section V presents a synthetic computational study, in which two rounding heuristics are evaluated. Our results indicate that high-quality solutions can be obtained even without resource augmentations. In particular, our heuristical rounding algorithm achieved 73.8% of the baseline's profit on average.

II. The Classic Multi-Commodity Formulation for the VNEP and its Limitations

In this section, we study the relaxation of the standard multi-commodity flow (MCF) formulation for the VNEP (cf. [2], [7]). We first show the positive result that the formulation is sufficiently strong to decompose virtual networks being *trees* into convex combinations of valid mappings. Subsequently, we show that the formulation fails to allow for the decomposition of *cyclic requests*. This not only impacts its applicability for randomized rounding but renders the formulation useless for approximations in general: it can be shown that the formulation's integrality gap is unbounded (cf. [17]).

A. The Classic Multi-Commodity Formulation

The classic MCF formulation for the VNEP is presented as Formulation 1 . We first describe its integer variant, which computes a single valid mapping for each request by using binary variables. The Linear Programming variant is obtained by relaxing the binary variables' domain to $[0,1]$.

The variable $x_r \in \{0,1\}$ indicates whether request $r \in \mathcal{R}$ is embedded or not. The variable $y^u_{r,i} \in \{0,1\}$ indicates whether virtual node $i \in V_r$ was mapped on substrate node $u \in V_S$. Similarly, the flow variable $z^{u,v}_{r,i,j} \in \{0,1\}$ indicates whether the substrate edge $(u,v) \in E_S$ is used to realize the virtual edge $(i,j) \in E_r$. The variable $a^{x,y}_r \geq 0$ denotes the cumulative allocations of request $r \in \mathcal{R}$ induced on resource $(x,y) \in R_S$.

By Constraint 2, the virtual node $i \in V_r$ of request $r \in \mathcal{R}$ must be placed on any of the suitable substrate nodes in $V^{r,i}_S$ iff. $x_r = 1$ holds and Constraint 3 forbids the mapping on nodes which may not host node i. Constraint 4 induces an unsplittable unit flow for each virtual edge $(i,j) \in E_r$ from the substrate location to which i was mapped to the substrate location to which j was mapped. By Constraint 5 virtual edges may only be mapped on *allowed* substrate edges. Constraints 6 and 7 compute the cumulative allocations and Constraint 8 guarantees that the substrate resource capacities are respected. The following lemma states the connectivity property enforced by Formulation 1 (see [17] for the proof).

Formulation 1: Classic MCF Formulation for the VNEP

$$\max \sum_{r \in \mathcal{R}} b_r x_r \tag{1}$$

$$\sum_{u \in V^{r,i}_S} y^u_{r,i} = x_r \qquad \forall r \in \mathcal{R}, i \in V_r \tag{2}$$

$$\sum_{u \in V_S \setminus V^{r,i}_S} y^u_{r,i} = 0 \qquad \forall r \in \mathcal{R}, i \in V_r \tag{3}$$

$$\begin{bmatrix} \sum\limits_{(u,v)\in\delta^+(u)} z^{u,v}_{r,i,j} \\ - \sum\limits_{(v,u)\in\delta^-(u)} z^{v,u}_{r,i,j} \end{bmatrix} = \begin{bmatrix} y^u_{r,i} \\ -y^u_{r,j} \end{bmatrix} \forall \begin{bmatrix} r \in \mathcal{R}, (i,j) \in E_r, \\ u \in V_S \end{bmatrix} \tag{4}$$

$$z^{u,v}_{r,i,j} = 0 \qquad \forall \begin{bmatrix} r \in \mathcal{R}, (i,j) \in E_r, \\ (u,v) \in E_S \setminus E^{r,i,j}_S \end{bmatrix} \tag{5}$$

$$\sum_{i \in V_r, \tau_r(i)=\tau} d_r(i) \cdot y^u_{r,i} = a^{\tau,u}_r \qquad \forall r \in \mathcal{R}, (\tau,u) \in R^V_S \tag{6}$$

$$\sum_{(i,j)\in E_r} d_r(i,j) \cdot z^{u,v}_{r,i,j} = a^{u,v}_r \qquad \forall r \in \mathcal{R}, (u,v) \in E_S \tag{7}$$

$$\sum_{r \in \mathcal{R}} a^{x,y}_r \leq d_S(x,y) \; \forall (x,y) \in R_S \tag{8}$$

Lemma 5 (Local Connectivity Property of Formulation 1)**.** *For any virtual edge $(i,j) \in E_r$ and any substrate node $u \in V^{r,i}_S$ with $y^u_{r,i} > 0$, there exists a path $P^{u,v}_{r,i,j}$ in G_S from u to $v \in V^{r,j}_S$ with $y^v_{r,j} > 0$, such that the flow along any edge of $P^{u,v}_{r,i,j}$ with respect to the variables $z^{\cdot,\cdot}_{r,i,j}$ is greater 0.*

The path $P^{u,v}_{r,i,j}$ can be computed in polynomial time.

B. Decomposing Solutions for Tree Requests

Given Lemma 5, we now present Algorithm 1 to decompose solutions to the LP Formulation 1 into convex combinations of valid mappings $\mathcal{D}_r = \{(f^k_r, m^k_r) | m^k_r \in \mathcal{M}_r, f^k_r > 0\}$ (cf. Section I-C), *if* the request's underlying undirected graph is a *tree*. Recall that in the LP formulation the binary variables are relaxed to take any value in the interval $[0,1]$.

Given a request $r \in \mathcal{R}$, the algorithm processes all virtual edges according to an arbitrary acyclic representation $G^{\mathcal{A}}_r = (V_r, E^{\mathcal{A}}_r, r_r)$ of the undirected interpretation of G_r being rooted at $r_r \in V_r$. Concretely, the edge set $E^{\mathcal{A}}_r$ is obtained from E_r by reorienting (some of the) edges, such that any node $i \in V_r$ can be reached from r_r. Considering tree requests for now, $G^{\mathcal{A}}_r$ is an arborescence and can be computed by a simple graph search of the underlying undirected graph starting at r_r. We denote by $\overleftarrow{E}^{\mathcal{A}}_r = E_r \setminus E^{\mathcal{A}}_r$ the edges whose orientations were reversed in the process of computing $G^{\mathcal{A}}_r$.

The algorithm extracts mappings m^k_r of value f^k_r iteratively, as long as $x_r > 0$ holds. Initially, in the k-th iteration, none of the virtual nodes and edges are mapped. As $x_r > 0$ holds, there must exist a node $u \in V^{r,r_r}_S$ with $y^{r_r}_{r,i} > 0$ by Constraint 2 and the algorithm accordingly sets $m^V_r(r_r) = u$. Given this initial fixing, the algorithm iteratively extracts nodes from the queue $\mathcal{Q}$ which have been already mapped and considers all outgoing virtual edges $(i,j) \in E^{\mathcal{A}}_r$. If an outgoing edge (i,j) is contained in E_r, Lemma 5 can be readily applied to obtain a joint mapping of the edge (i,j) and its head j. If the edge's orientation was reversed, i.e. if $(i,j) \in \overleftarrow{E}^{\mathcal{A}}_r$ holds, Lemma 5 is applied *while reversing* the flow's direction (see Lines 13-16).

First, note that by the repeated application of Lemma 5, the mapping of virtual nodes and edges is valid. As $G^{\mathcal{A}}_r$ is an arborescence, each edge and each node of $G^{\mathcal{A}}_r$ will eventually be mapped and hence m^k_r is a valid mapping. The mapping value f^k_r is computed as the minimum of the mapping variables $\mathcal{V}_k$ used for constructing m^k_r. Reducing the values of the mapping variables together with the allocation variables $\vec{a}_r$ (Lines 20-21), the Constraints 2-7 continue to hold.

As the decomposition process continues as long as $x_r > 0$ holds and in the k-th step at least one variable's value is set to 0, the algorithm terminates with a complete decomposition

Algorithm 1: Decompositioning MCF solutions for Tree Requests

```
Input  : Tree request $r \in \mathcal{R}$, solution $(x_r, \vec{y}_r, \vec{z}_r, \vec{a}_r)$ to LP
         Formulation 1, acyclic reorientation $G_r^{\mathcal{A}} = (V_r, E_r^{\mathcal{A}}, r_r)$
Output: Convex combination $\mathcal{D}_r = \{D_r^k = (f_r^k, m_r^k)\}_k$

set $\mathcal{D}_r \leftarrow \emptyset$ and $k \leftarrow 1$
while $x_r > 0$ do
    set $m_r^k = (m_r^V, m_r^E) \leftarrow (\emptyset, \emptyset)$
    set $\mathcal{Q} = \{r_r\}$
    choose $u \in V_S^{r,r_r}$ with $y_{r,r_r}^u > 0$ and set $m_r^V(r_r) \leftarrow u$
    while $|\mathcal{Q}| > 0$ do
        choose $i \in \mathcal{Q}$ and set $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{i\}$
        foreach $(i,j) \in E_r^{\mathcal{A}}$ do
            if $(i,j) \in E_r$ then
                compute $\overrightarrow{P}_{r,i,j}^{u,v}$ connecting $m_r^V(i) = u$ to $v \in V_S^{r,j}$
                        according to Lemma 5
                set $m_r^V(j) = v$ and $m_r^E(i.j) = \overrightarrow{P}_{r,i,j}^{u,v}$
            else
                let $\overleftarrow{z}_{r,i,j}^{v',u'} \triangleq z_{r,j,i}^{u',v'}$ for all $(u',v') \in E_S$
                compute $\overleftarrow{P}_{r,i,j}^{v,u}$ connecting $m_r^V(i) = v$ to $u \in V_S^{r,j}$
                        according to Lemma 5
                set $\overrightarrow{P}_{r,j,i}^{u,v} = \texttt{reverse}(\overleftarrow{P}_{r,i,j}^{v,u})$
                set $m_r^V(i) = u$ and $m_r^E(j,i) = \overrightarrow{P}_{r,j,i}^{u,v}$
            set $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{j\}$
    set $\mathcal{V}_k \leftarrow \left( \{x_r\} \cup \{y_{r,i}^{m_r^V(i)} | i \in V_r\} \cup \{z_{r,i,j}^{u,v} | (i,j) \in E_r, (u,v) \in m_r^E(i,j)\} \right)$
    set $f_r^k \leftarrow \min \mathcal{V}_k$
    set $v \leftarrow v - f_r^k$ for all $v \in \mathcal{V}_k$
    set $a_r^{x,y} \leftarrow a_r^{x,y} - f_r^k \cdot A(m_r^k, x, y)$ for all $(x,y) \in R_S$
    add $D_r^k = (f_r^k, m_r^k)$ to $\mathcal{D}_r$ and set $k \leftarrow k+1$
return $\mathcal{D}_r$
```

for which $\sum_k f_r^k = x_r$ holds. Furthermore, the algorithm has polynomial runtime, as in each iteration at least one variable is set to 0 and the number of variables for request r is bounded by $\mathcal{O}(|E_r| \cdot |E_S|)$. Hence, we obtain the following:

Lemma 6. *Given a virtual network request $r \in \mathcal{R}$, whose underlying undirected graph is a tree, Algorithm 1 decomposes a solution $(x_r, \vec{y}_r, \vec{z}_r, \vec{a}_r)$ to the LP Formulation 1 into valid mappings $\mathcal{D}_r = \{(m_r^k, f_r^k)\}_k$, such that the following holds:*

- *The decomposition is complete, i.e. $x_r = \sum_k f_r^k$ holds.*
- *The decomposition's resource allocations are bounded by $\vec{a}_r$: $a_r^{x,y} \geq \sum_k f_r^k \cdot A(m_r^k, x, y)$ holds for $(x,y) \in R_S$.*

C. Limitations of the Classic MCF Formulation

Above it was shown that LP solutions to the classic MCF formulation can be decomposed into convex combinations of valid mappings *if* the underlying graph is a tree. This does not hold anymore when considering cyclic virtual networks:

Theorem 7. *Solutions to the standard LP Formulation 1 can in general not be decomposed into convex combinations of valid mappings if the virtual networks contain cycles.*

Proof. In Figure 2 we visually depict an example of a solution to the LP Formulation 1 from which *not a single* valid mapping can be extracted. The validity of the depicted solution follows from the fact that the virtual node mappings sum to 1 and

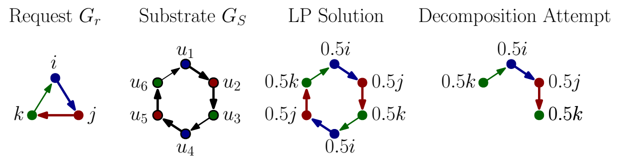

Fig. 2. Example showing that solutions to the LP Formulation 1 can in general not be decomposed into convex combinations of valid mappings. Request r is a simple cyclic graph which shall be mapped on the substrate graph G_S. We assume following node mapping restrictions $V_S^{r,i} = \{u_1, u_4\}$, $V_S^{r,j} = \{u_2, u_5\}$, $V_S^{r,k} = \{u_3, u_6\}$. The LP solution with $x_r = 1$ is depicted as follows. Substrate nodes are annotated with the mapping of virtual nodes. Hence, $0.5i$ at node u_1 indicates $y_{r,i}^{u_1} = 1/2$, i.e. that virtual node i is mapped with 0.5 on substrate node u_1. Substrate edges are colored according to the color of virtual links mapped onto it. Virtual links are all mapped using flow values $1/2$. Accordingly, for example $z_{r,i,j}^{u_1,u_2} = 1/2$ holds.

each virtual node connects to its neighboring node with half a unit of flow. Assume for the sake of contradiction that the depicted solution can be decomposed. As virtual node $i \in V_r$ is mapped onto substrate node $u_1 \in V_S$, and $u_2 \in V_S$ is the only neighboring node with respect to variables $z_{r,i,j}$ that hosts $j \in V_r$, there must exist a mapping (m_r^V, m_r^E) with $m_r^V(i) = u_1$ and $m_r^V(j) = u_2$. Similarly, $m_r^V(k) = u_3$ must hold. However, for $m_r^V(i) = u_1$, the virtual node k must be mapped to u_6, as otherwise the embedding of (k,i) cannot lead to substrate node u_1. Hence the virtual node $k \in V_r$ must be mapped both on u_6 and u_3. As this is not possible, and the same argument holds when considering the mapping of i onto u_4, no valid mapping can be extracted. □

This non-decomposability also induces large integrality gaps, as proven in our extended technical report [17].

Theorem 8. *The integrality gap of the MCF formulation is unbounded. This even holds under infinite substrate capacities.*

III. Novel Decomposable LP Formulation

In this section, we present a novel LP formulation and its accompanying decomposition algorithm for the class of cactus request graphs, i.e. graphs for which cycles intersect in at most a single node (in its undirected interpretation). Accordingly, these graphs can be uniquely decomposed into cycles and a single forest (cf. Lemma 9 below).

Before delving into the details of our novel LP formulation, we discuss our main insight on how to overcome the limitations of the MCF formulation and accordingly how to derive decomposable formulations. To this end, it is instructive, to revisit the non-decomposable example of Figure 2 by applying the decomposition Algorithm 1 on the depicted LP solution. Concretely, we consider the acyclic reorientation $G_r^{\mathcal{A}} = (V_r, E_r^{\mathcal{A}}, r_r)$ with $E_r^{\mathcal{A}} = \{(i,k), (i,j), (j,k)\}$, such that i is the root, $r_r = i$. Assuming that i is initially mapped on node u_1, Algorithm 1 will map edges (i,k) and (i,j) first, setting $m_r^V(k) = u_6$ and $m_r^V(j) = u_2$ However, when the edge (j,k) is processed, k must be mapped on substrate node $u_3 \neq m_r^V(k)$ and the algorithm hence fails to produce a valid mapping. Accordingly, to avoid such *diverging* node mappings, our key idea is to decide the mapping location of nodes with more than one incoming edge (with respect to the request's acyclic reorientation) *a priori*.

By considering only cactus request graphs, this can be implemented rather easily as exactly one node of each cycle has more than one incoming edge: one only needs to ensure *compatibility* of node mappings for this node. To resolve potential conflicts for the mapping of this unique *cycle target*, our formulation employs *multiple* copies of the MCF formulation for the respective cycle subgraph. Specifically, considering a cycle with virtual *target* node k, we instantiate one MCF formulation per substrate node $w \in V_S^{r,k}$ onto which k can be mapped. Accordingly, this yields at most $|V_S|$ many copies and for each of these copies k is *fixed* to one specific (substrate) mapping location. Accordingly, as the mapping location of k is fixed to a specific node, valid mappings for the respective cycles can always be extracted from such a MCF copy: the mappings of k cannot possibly diverge.

A. Cactus Request Graph Decomposition and Notation

We decompose cactus request graphs as follows (cf. [17]).

Lemma 9. *Consider a cactus request graph G_r and its acyclic reorientation $G_r^{\mathcal{A}}$ of G_r. The graph $G_r^{\mathcal{A}}$ can be uniquely partitioned into subgraphs $\{G_r^{\mathcal{A},C_1}, \dots, G_r^{\mathcal{A},C_n}\} \sqcup G_r^{\mathcal{A},\mathcal{F}}$, s.t.:*

1) *The subgraphs $\{G_r^{\mathcal{A},C_1}, \dots, G_r^{\mathcal{A},C_n}\}$ correspond to the (undirected) cycles of G_r and $G_r^{\mathcal{A},\mathcal{F}}$ is the* forest *remaining after removing the cyclic subgraphs. We denote the index set of the cycles by $\mathcal{C}_r = \{C_1, \dots, C_n\}$.*
2) *The subgraphs partition the edges of $E_r^{\mathcal{A}}$: an edge $(i,j) \in E_r^{\mathcal{A}}$ is contained in exactly one of the subgraphs.*
3) *The edge set $E_r^{\mathcal{A},C_k}$ of each cycle $C_k \in \mathcal{C}_r$ can itself be partitioned into two branches $\mathcal{B}_1^{C_k}$ and $\mathcal{B}_2^{C_k}$, such that both lead from $s_r^{C_k} \in V_r^{\mathcal{A},C_k}$ to $t_r^{C_k} \in V_r^{\mathcal{A},C_k}$.*

Additionally, we denote by $G_r^{C_k}$ and $G_r^{\mathcal{F}}$ the subgraphs that agree with E_r on the edge orientations and use $V_{S,t}^{C_k} = V_S^{r,t_r^{C_k}}$ to denote the substrate nodes on which $t_r^{C_k}$ can be mapped.

B. Novel LP Formulation for Cactus Requests

Our novel Formulation 2 uses the a priori partition of $G_r^{\mathcal{A}}$ into cycles $G_r^{\mathcal{A},C_k}$ and the forest $G_r^{\mathcal{A},\mathcal{F}}$ to construct MCF formulations for the respective subgraphs: for the subgraph $G_r^{\mathcal{F}}$ a single copy is used (cf. Constraint 10) while for the cyclic subgraphs a single MCF formulation is employed *per* potential target location $V_{S,t}^{C_k}$ (cf. Constraint 11). We index the variables of these sub-LPs by employing square brackets.

To bind together these (at first) independent MCF formulations, we reuse the variables $\vec{x}, \vec{y}$, and $\vec{a}$ introduced already for the MCF formulation. We refer to these variables, which are defined outside of the sub-LP formulations, as *global variables* and do not index these. As we only consider the LP formulation, all variables are continuous.

The different sub-formulations are linked as follows. We employ Constraint 12 to enforce the setting of the (global) node mapping variables (cf. Constraint 2 of Formulation 1). By Constraints 13 and 14, the node mappings of the sub-LPs for mapping the subgraphs must agree with the global node mapping variables. With respect to cyclic subgraphs, we note that Constraint 14 allows for distributing the global node mappings to any of the $|V_{S,t}^{C_k}|$ formulations: only the sum of the node mapping variables must agree with the global node mapping variable. Constraint 15 is of crucial importance for the decomposability: considering the sub-LP for cycle C_k and target node $w \in V_{S,t}^{C_k}$, it enforces that the target node $t_r^{C_k}$ of the cycle C_k *must* be mapped on w. Thus, in the sub-LP $[C_k, w]$ both branches $\mathcal{B}_1^{C_k}$ and $\mathcal{B}_2^{C_k}$ of cycle C_k are *pre-determined* to lead to the node w. Lastly, for computing node allocations the global node mapping variables are used (cf. Constraint 16) and for computing edge allocations the sub-LP formulations' allocations are considered (cf. Constraint 17).

C. Decomposing Solutions to the Novel LP Formulation

We now show how to adapt the decomposition Algorithm 1 to decompose solutions to Formulation 2.

To decompose the LP solution for a request r the acyclic reorientation $G_r^{\mathcal{A}}$, which was also used for constructing the LP, must be handed over to the decomposition algorithm.

As the novel LP formulation does not contain (global) edge mapping variables, the edge mapping variables used in Lines 10 and 13 of Algorithm 1 must be substituted by edge mapping variables of the respective sub-LP formulations. Concretely, as each edge of the request graph G_r is covered exactly once, it is clear whether a virtual edge $(i,j) \in E_r$ is part of $G_r^{\mathcal{F}}$ or a cyclic subgraph $G_r^{C_k}$. If $(i,j) \in G_r^{\mathcal{F}}$ holds, then the edge mapping variables $z_{r,i,j}^{\cdot,\cdot}[\mathcal{F}_r]$ are used. If on the other hand the edge $(i,j) \in E_r$ is covered in the cyclic subgraph $G_r^{C_k}$, then there exist $|V_{S,t}^{C_k}|$ many sub-LPs to choose the respective edge mapping variables from. To ensure the decomposability, we proceed as follows.

If the edge $(i,j) \in E_r^{\mathcal{A}}$ is the first edge of $G_r^{C_k}$ to be mapped in the k-th iteration, the mapping variables $z_{r,i,j}^{\cdot,\cdot}[C_k, w]$ be-

Formulation 2: Novel LP for Cactus Requests

$$\max \sum_{r \in \mathcal{R}} b_r x_r \tag{9}$$

$$\text{Cons. (2) - (7) for } G_r^{\mathcal{F}} \text{ on variables } (x_r, \vec{y}_r, \vec{z}_r, \vec{a}_r)[\mathcal{F}_r] \qquad \forall r \in \mathcal{R} \tag{10}$$

$$\text{Cons. (2) - (7) for } G_r^{C_k} \text{ on variables } (x_r, \vec{y}_r, \vec{z}_r, \vec{a}_r)[C_k, w] \qquad \forall r \in \mathcal{R}, C_k \in \mathcal{C}_r, w \in V_{S,t}^{C_k} \tag{11}$$

$$x_r = \sum_{u \in V_S^{r,i}} y_{r,i}^u \qquad \forall r \in \mathcal{R}, i \in V_r \tag{12}$$

$$y_{r,i}^u = y_{r,i}^u[\mathcal{F}] \qquad \forall r \in \mathcal{R}, i \in V_r^{\mathcal{F}}, u \in V_S^{r,i} \tag{13}$$

$$y_{r,i}^u = \sum_{w \in t_r^{C_k}} y_{r,i}^u[C_k, w] \qquad \forall \begin{bmatrix} r \in \mathcal{R}, i \in V_r, u \in V_S^{r,i}, \\ C_k \in \mathcal{C}_r : i \in V_r^{C_k} \end{bmatrix} \tag{14}$$

$$0 = y_{r,t_r^{C_k}}^u[C_k, w] \qquad \forall \begin{bmatrix} r \in \mathcal{R}, C_k \in \mathcal{C}_r, w \in V_{S,t}^{C_k}, \\ u \in V_{S,t}^{C_k} \setminus \{w\} \end{bmatrix} \tag{15}$$

$$a_r^{\tau,u} = \sum_{i \in V_r, \tau_r(i) = \tau} d_r(i) \cdot y_{r,i}^u \qquad \forall r \in \mathcal{R}, (\tau, u) \in R_S^V \tag{16}$$

$$a_r^{u,v} = a_r^{u,v}[\mathcal{F}] + \sum_{C_k \in \mathcal{C}_r, w \in V_{S,t}^{C_k}} a_r^{u,v}[C_k, w] \qquad \forall r \in \mathcal{R}, (u,v) \in E_S \tag{17}$$

$$\sum_{r \in \mathcal{R}} a_r^{x,y} \leq d_S(x,y) \qquad \forall (x,y) \in R_S \tag{18}$$

longing to an arbitrary target node w, with $y_{r,i}^{m_r^V(i)}[C_k, w] > 0$, are used. Such a node w exists by Constraint 14.

If another edge (i', j') of the same cycle was already mapped in the k-th iteration, the *same* sub-LP as chosen before is considered. Accordingly, the mapping of cycle target nodes cannot conflict and as these are the only nodes with potential mapping conflicts, the returned mappings are always valid.

To successfully iterate the extraction process, the steps taken in Lines 18 - 21 of Algorithm 1 must be adapted to consider the sub-LP variables. Again, as in each iteration at least a single variable of the LP is set to 0 and as the novel Formulation 2 contains at most $\mathcal{O}(|V_S|)$ times more variables than the MCF Formulation 1, the decomposition algorithm still runs in polynomial-time. Hence, we conclude that the result of Lemma 6 carries over to the novel LP Formulation 2 for *cactus* request graphs and state the following theorem.

Theorem 10. *Given a solution $(x_r, \vec{y}_r, \vec{z}_r, \vec{a}_r)$ to the novel LP Formulation 2 for a cactus request graph G_r, the solution can be decomposed into a convex combination of valid mappings $\mathcal{D}_r = \{(m_r^k, f_r^k)\}_k$ in polynomial-time, such that:*

- *The decomposition is complete, i.e. $x_r = \sum_k f_r^k$ holds.*
- *The decomposition's resource allocations are bounded by $\vec{a}_r$: $a_r^{x,y} \geq \sum_k f_r^k \cdot A(m_r^k, x, y)$ holds for $(x, y) \in R_S$.*

IV. Approximation via Randomized Rounding

Above we have shown how optimal convex combinations for the VNEP can be computed for cactus requests. Given these convex combinations, the pseudo-code of our approximation for the VNEP is presented as Algorithm 2.

The algorithm first performs a preprocessing in Lines 1-3 by removing all requests which cannot be fully (fractionally) embedded in the absence of other requests, as these can never be part of any feasible solution. In Lines 4-6 an optimal solution to the novel LP Formulation 2 is computed and afterwards decomposed into convex combinations. Then, in Lines 7-9, the rounding is performed: for each request r a mapping m_r^k is selected with probability f_r^k. Importantly, the sum of probability may not sum to 1, i.e. with probability $1 - \sum_k f_r^k$ the request r is not embedded.

The rounding procedure is iterated as long as the constructed solution is not of sufficient quality or until the number of maximal rounding tries is exceeded. Concretely, we seek (α, β, γ)-approximate solutions which achieve at least a factor of $\alpha \leq 1$ times the optimal (LP) profit and exceed node and edge capacities by at most factors of $\beta \geq 1$ and $\gamma \geq 1$, respectively. In the following we derive parameters α, β, and γ for which solutions can be found *with high probability*.

Algorithm 2: Randomized Rounding for the VNEP

```
1 foreach $r \in \mathcal{R}$ do              // preprocess requests
2     compute LP Formulation 2 for request $r$ maximizing $x_r$
3     if $x_r < 1$ then remove request $r$ from the set $\mathcal{R}$
4 compute LP Formulation 2 for $\mathcal{R}$ maximizing $\sum_{r \in \mathcal{R}} b_r \cdot x_r$
5 foreach $r \in \mathcal{R}$ do              // perform decomposition
6     compute $\mathcal{D}_r = \{(f_r^k, m_r^k)\}_k$ from LP solution
7 do                                        // perform randomized rounding
8     foreach $r \in \mathcal{R}$ select $m_r^k$ with probability $f_r^k$
9 while ( solution is not $(\alpha, \beta, \gamma)$-approximate and
          maximal rounding tries are not exceeded )
```

Note that Algorithm 2 is indeed a polynomial-time algorithm, as the size of the novel LP Formulation 2 is polynomially bounded and can hence be solved in polynomial-time.

A. Probabilistic Guarantee for the Profit

For bounding the profit achieved by the randomized rounding scheme, we recast the profit achieved in terms of random variables. The *discrete* random variable $Y_r \in \{0, b_r\}$ models the profit achieved by the rounding of request $r \in \mathcal{R}$. According to our rounding scheme, we have $\mathbb{P}(Y_r = b_r) = \sum_k f_r^k$ and $\mathbb{P}(Y_r = 0) = 1 - \sum_k f_r^k$. We denote the overall profit by $B = \sum_{r \in \mathcal{R}} Y_r$ with $\mathbb{E}(B) = \sum_{r \in \mathcal{R}} b_r \cdot \sum_k f_r^k$. Denoting the profit of an optimal LP solution by B_{LP}, we have $\mathrm{B}_{\mathrm{LP}} = \mathbb{E}(B)$ due to the decomposition's completeness (cf. Theorem 10).

By preprocessing the requests and confirming that each request can be fully embedded, the LP will attain at least the maximal profit of any of the considered requests:

Lemma 11. $\mathbb{E}(B) = \mathrm{B}_{\mathrm{LP}} \geq \max_{r \in \mathcal{R}} b_r$ *holds.*

We employ the following Chernoff bound over continuous variables to bound the probability of achieving a small profit.

Theorem 12 (Chernoff Bound [19])**.** *Let $X = \sum_{i=1}^n X_i$, $X_i \in [0, 1]$, be a sum of n independent random variables. For any $0 < \varepsilon < 1$, the following holds:*

$$\mathbb{P}\big(X \leq (1 - \varepsilon) \cdot \mathbb{E}(X)\big) \leq \exp(-\varepsilon^2 \cdot \mathbb{E}(X)/2)$$

Theorem 13. *Let B_{IP} denote the profit of an optimal solution. Then $\mathbb{P}(B < 1/3 \cdot \mathrm{B}_{\mathrm{IP}}) \leq \exp(-2/9) \approx 0.8007$ holds.*

Proof. Let $\hat{b} = \max_{r \in \mathcal{R}} b_r$ be the maximum benefit among the pre-processed requests. We consider random variables $Y'_r = Y_r / \hat{b}$, such that $Y'_r \in [0, 1]$ holds. Let $B' = \sum_{r \in \mathcal{R}} Y'_r = B / \hat{b}$.

As $\mathbb{E}(B) = \mathrm{B}_{\mathrm{LP}} \geq \hat{b}$ holds (cf. Lemma 11), we have $\mathbb{E}(B') \geq 1$. Choosing $\varepsilon = 2/3$ and applying Theorem 12 on B' we obtain $\mathbb{P}\big(B' \leq (1/3) \cdot \mathbb{E}(B')\big) \leq \exp(-2 \cdot \mathbb{E}(B')/9)$. Plugging in the *minimal* value of $\mathbb{E}(B')$, i.e. 1, into the equation we obtain: $\mathbb{P}\big(B' \leq (1/3) \cdot \mathbb{E}(B')\ \big) \leq \exp(-2/9)$ and by linearity $\mathbb{P}\big(B \leq (1/3) \cdot \mathbb{E}(B)\ \big) \leq \exp(-2/9)$.

Denoting the profit of an optimal solution by B_{IP} and observing that $\mathrm{B}_{\mathrm{IP}} \leq \mathrm{B}_{\mathrm{LP}}$ holds as the linear relaxation yields an upper bound, we have $\mathrm{B}_{\mathrm{IP}}/3 \leq \mathbb{E}(B)/3$. Accordingly, we conclude that, $\mathbb{P}\big(B \leq (1/3) \cdot \mathrm{B}_{\mathrm{IP}}\big) \leq \exp(-2/9)$ holds. □

B. Probabilistic Guarantee for Resource Augmentations

In the following, we analyze the probability that a rounded solution exceeds substrate capacities by a certain factor.

We first note that $d_{\max}(r, x, y) \leq d_S(x, y)$ holds for all resources $(x, y) \in R_S$ and all requests $r \in \mathcal{R}$. We model the allocations on resource $(x, y) \in R_S$ by request $r \in \mathcal{R}$ as random variable $A_{r,x,y} \in [0, A_{\max}(r, x, y)]$. By definition, we have $\mathbb{P}(A_{r,x,y} = A(m_r^k, x, y)) = f_r^k$ and $\mathbb{P}(A_{r,x,y} = 0) = 1 - \sum_k f_r^k$. Furthermore, we denote by $A_{x,y} = \sum_{r \in \mathcal{R}} A_{r,x,y}$ the random variable capturing the overall allocations on resource $(x, y) \in R_S$.

$\mathbb{E}(A_{x,y}) = \sum_{r\in\mathcal{R}}\sum_k f_r^k \cdot A(r,x,y)$ holds by Theorem 10, we obtain $\mathbb{E}(A_{x,y}) \leq d_S(x,y)$ for all resources $(x,y) \in R_S$.

We employ Hoeffding's inequality to upper bound $A_{x,y}$.

Theorem 14 (Hoeffding's inequality [19]). *Let $X = \sum_{i=1}^n X_i$, $X_i \in [a_i, b_i]$, be a sum of n independent random variables. The following holds for any $t \geq 0$:*

$$\mathbb{P}(X - \mathbb{E}(X) \geq t) \leq \exp(-2t^2/(\sum\nolimits_i (b_i - a_i)^2))$$

Lemma 15. *Consider a resource $(x,y) \in R_S$ and $0 < \varepsilon \leq 1$, such that $d_{\max}(r,x,y)/d_S(x,y) \leq \varepsilon$ holds for $r \in \mathcal{R}$. Let $\Delta(x,y) = \sum_{r\in\mathcal{R}:d_{\max}(r,x,y)>0}(A_{\max}(r,x,y)/d_{\max}(r,x,y))^2$.*

$$\mathbb{P}(A_{x,y} \geq \delta(\lambda)\cdot d_S(x,y)) \leq \lambda^{-4} \tag{19}$$

holds for $\delta(\lambda) = 1 + \varepsilon \cdot \sqrt{2\cdot\Delta(x,y)\cdot\log(\lambda)}$ and any $\lambda > 0$.

Proof. We apply Hoeffding with $t = (1-\delta(\lambda))\cdot d_S(x,y)$:

$$\begin{aligned}
&\mathbb{P}(A_{x,y} - \mathbb{E}(A_{x,y}) \geq (1-\delta(\lambda))\cdot d_S(x,y)) \\
&\leq \exp\Big(\frac{-4\cdot\varepsilon^2\cdot\log(\lambda)\cdot\Delta(x,y)\cdot d_S^2(x,y)}{\sum\limits_{r\in\mathcal{R}}(A_{\max}(r,x,y))^2}\Big) \\
&\leq \exp\Big(\frac{-4\cdot\varepsilon^2\cdot\log(\lambda)\cdot\Delta(x,y)\cdot d_S^2(x,y)}{\sum\limits_{r\in\mathcal{R}:d_{\max}(r,x,y)>0}(A_{\max}(r,x,y))^2}\Big) \\
&\leq \exp\Big(\frac{-4\cdot\varepsilon^2\cdot\log(\lambda)\cdot\Delta(x,y)\cdot d_S^2(x,y)}{\sum\limits_{r\in\mathcal{R}:d_{\max}(r,x,y)>0}(\varepsilon\cdot d_S(x,y)\cdot A_{\max}(r,x,y)/d_{\max}(r,x,y))^2}\Big) \\
&\leq \exp\Big(\frac{-4\cdot\log(\lambda)\cdot\Delta(x,y)}{\sum\limits_{r\in\mathcal{R}:d_{\max}(r,x,y)>0}(A_{\max}(r,x,y)/d_{\max}(r,x,y))^2}\Big) = \lambda^{-4}
\end{aligned}$$

The second inequality holds, as $A_{\max}(r,x,y) > 0$ implies $d_{\max}(r,x,y) > 0$. For the third inequality, $A_{\max}(r,x,y) \leq \varepsilon\cdot d_S(x,y)\cdot A_{\max}(r,x,y)/d_{\max}(r,x,y)$ is used, which follows from the assumption $d_{\max}(r,x,y) \leq \varepsilon\cdot d_S(x,y)$ and $d_{\max}(r,x,y) > 0$. In the next step, $\varepsilon^2\cdot d_S^2(x,y)$ is reduced from the fraction. As the denominator equals $\Delta(x,y)$ by definition, the final equality follows. Lastly, we utilize that the expected allocation $\mathbb{E}(A_{x,y})$ is upper bounded by the resource's capacity $d_S(x,y)$ to obtain Equation 19. ☐

Given Lemma 15, we obtain the following corollary.

Corollary 16. *Let $\varepsilon \leq 1$ be chosen minimally, such that $d_{\max}(r,x,y)/d_S(x,y) \leq \varepsilon$ holds for all resources $(x,y) \in R_S$ and all requests $r \in \mathcal{R}$. Let $\Delta(X) = \max_{(x,y)\in X}\Delta(x,y)$,*

$$\begin{aligned}
\beta &= (1 + \varepsilon\cdot\sqrt{2\cdot\Delta(R_S^V)\cdot\log(|V_S|\cdot|\mathcal{T}|)})\text{ , and} \\
\gamma &= (1 + \varepsilon\cdot\sqrt{2\cdot\Delta(E_S)\cdot\log(|E_S|)})\,.
\end{aligned}$$

The following holds for all node resources $(\tau,u) \in R_S^V$ and edge resources $(u,v) \in E_S$, respectively:

$$\mathbb{P}(A_{\tau,u} \geq \beta\cdot d_S(\tau,u)) \leq (|V_S|\cdot|\mathcal{T}|)^{-4} \tag{20}$$

$$\mathbb{P}(A_{u,v} \geq \gamma\cdot d_S(u,v)) \leq |E_S|^{-4} \tag{21}$$

Proof. First, note that ε is chosen over all resources and requests and that $\Delta(R_S^V) \geq \Delta(\tau,u)$ and $\Delta(E_S) \geq \Delta(u,v)$ hold for $(\tau,u) \in R_S^V$ and $(u,v) \in E_S$, respectively. Equations 20 and 21 are then obtained from Lemma 15 by setting $\lambda = |V_S|\cdot|\mathcal{T}|$ for nodes and $\lambda = |E_S|$ for edges. ☐

C. Approximation Result

Given the probabilistic bounds established above, the main approximation result is obtained via a union bound.

Theorem 17. *Assume $|V_S| \geq 3$. Let β and γ be defined as in Corollary 16. Algorithm 2 returns (α,β,γ)-approximate solutions for the VNEP (restricted on cactus request graphs) of at least an $\alpha = 1/3$ fraction of the optimal profit, and allocations on nodes and edges within factors of β and γ of the original capacities, respectively, with high probability.*

Proof. We employ the following union bound argument. Employing Corollary 16 and as there are at most $|V_S|\cdot|\mathcal{T}|$ node resources and at most $|V_S|^2$ edges, the *joint* probability that any resource exceeds their respective capacity by factors of β or γ is upper bounded by $(|V_S|\cdot|\mathcal{T}|)^3 + |V_S|^2 \leq 1/27 + 1/9$ for $|V_S| \geq 3$. By Theorem 13 the probability of *not* finding a solution achieving an $\alpha = 1/3$ fraction of the optimal objective is upper bounded by $\exp(-2/9)$. Hence, the probability to not find a (α,β,γ)-approximate solution within a single round is upper bounded by $\exp(-2/9) + 1/9 + 1/27 \leq 19/20$. The probability to return a suitable solution within $N \in \mathbb{N}$ rounding tries is lower bounded by $1-(19/20)^N$ and Algorithm 2 yields approximate solutions for the VNEP *with high probability*. ☐

D. Discussion & Proposed Heuristics

Theorem 17 yields the first approximation algorithm for the profit variant of the VNEP. However, the direct application of Algorithm 2 to compute (α,β,γ)-approximate solutions is made difficult by the cumbersome definition of the terms $\Delta(R_S^V)$ and $\Delta(E_S)$. Specifically, computing β and γ exactly requires enumerating all valid mappings, which is not feasible. Hence, to directly apply Algorithm 2, the respective values have to be estimated. Considering $\Delta(R_S^V)$, the following upper bound can be easily established: $\Delta(R_S^V) \leq |\mathcal{R}|\cdot\max_{r\in\mathcal{R}}|V_r|$. However, plugging this bound into the definition of β yields rather large resource violations of $\beta \in \mathcal{O}(\varepsilon\cdot\sqrt{|\mathcal{R}|\cdot\max_{r\in\mathcal{R}}|V_r|\cdot\log(|V_S|\cdot|\mathcal{T}|)})$.

To overcome estimating β and γ, we propose the following:

Vanilla Rounding: A fixed number of solutions is rounded at random as in Line 7 of Algorithm 2. Afterwards, the best solution is returned according to some metric. In particular, in Section V we study the metric returning the solution of highest profit among the solutions minimizing the maximal resource augmentation.

Heuristical Rounding: In most settings resource augmentations are to be avoided based on their negative impact on the customer's Quality-of-Service. Hence, to obtain solutions not violating any resource's capacity, we propose to adapt the rounding scheme by simply *discarding* selected mappings, whose addition would exceed resource capacities. To increase the diversity of found solutions, the order in which requests are processed is permuted before each rounding iteration.

V. Explorative Computational Study

We now complement our formal approximation result in the standard multi-criteria model with resource augmentation with an extensive computational study. Specifically, we study the performance of vanilla rounding and heuristical rounding without resource augmentations as introduced above.

As we are not aware of any systematic evaluation of the profit maximization in the offline settings, we present a synthetic but extensive computational study. Specifically, we have generated 1,500 offline VNEP instances with varying request numbers and varying demand-to-capacity ratios. For all instances, baseline solutions were computed by solving the Mixed-Integer Programming Formulation 1.

We restrict our discussion to our main results and refer the reader to our technical report at [17] for additional details. We have implemented all presented algorithms in Python 2.7 employing Gurobi 7.5.1 to solve Mixed-Integer Programs and Linear Programs. Our source code is freely available at [20]. All experiments were executed on a server equipped with Intel Xeon E5-4627v3 CPUs running at 2.6 GHz.

A. Instance Generation

We use the GÉANT topology[1] as substrate network. It consists of 40 nodes and 122 edges. We consider a single node type and set node and edge capacities uniformly to 100.

a) Request Topology Generation: Cactus graph requests are generated by (i) sampling a random binary tree of maximum depth 3, (ii) adding additional edges randomly as long as they do not refute the cactus property *as long as such edges exist*, and (iii) orienting edges arbitrarily.

We only consider requests containing at least 3 nodes. According to our generation parameters, the expected number of nodes and edges is 6.54 and 7.28, respectively. On average, 61% of the edges lie on a cycle.

b) Mapping Restrictions: To force the virtual networks to span across the whole substrate network, we restrict the mapping of virtual nodes to one quarter of the substrate nodes: each virtual node can be mapped on ten substrate nodes. The mapping of virtual edges is not restricted.

c) Demand Generation: We control the demand-to-capacity ratio of node and edge resource using a node resource factor NRF and an edge resource factor ERF. The request's demands are drawn from an exponential distribution and afterwards normalized, such that the following holds:

$$\sum_{r\in\mathcal{R}}\sum_{i\in V_r} d_r(i) = \mathrm{NRF}\cdot\sum_{u\in V_S} d_S(u)$$
$$\mathrm{ERF}\cdot\sum_{r\in\mathcal{R}}\sum_{(i,j)\in E_r} d_r(i,j) = \sum_{(u,v)\in E_S} d_S(u,v)$$

The resource factors can be best understood under the assumption that all requests are embedded. Under this assumption, a resource factor $\mathrm{NRF} = 0.6$ implies that the node load – averaged over all substrate nodes – equals exactly 60%. As virtual edges can be mapped on arbitrarily long paths (even of length 0), the edge resource factor should be understood as follows: the ERF equals 'the number of substrate edges that each virtual edge may use'. In particular, a factor $\mathrm{ERF} = 0.5$ implies that if *each* virtual edge spans *exactly* 0.5 substrate edges, then edge resource utilization equals exactly 100%.

[1]Obtained from http://www.topology-zoo.org/ (version March 2012).

Hence, while increasing the NRF renders node resources more scarce, increasing the ERF reduces edge resource scarcity.

d) Profit Computation: To correlate the profit of a request with its size, its resource demands, and its mapping restrictions, we compute for each request its minimal embedding costs as follows. The cost $c(u,v)$ of using an edge $(u,v)\in V_S$ equals the geographical distance of its endpoints. The cost of nodes is set uniformly to $c(\cdot,u) = \sum_{(u,v)\in E_S} c(u,v)/|V_S|$ for all $u \in V_S$. Hence, the total node cost equals the total edge cost. Defining the cost of a mapping m_r to be $\sum_{(x,y)\in R_S} A(m_r,x,y)\cdot c(x,y)$, we compute the minimum cost embedding for each request $r\in\mathcal{R}$ using an adaption of Mixed-Integer Program 1 and set b_r accordingly.

e) Parameter Space: We consider the following parameters $|\mathcal{R}| \in \{40,60,80,100\}$, $\mathrm{NRF}\in\{0.2,0.4,0.6,0.8,1.0\}$, $\mathrm{ERF}\in\{0.25,\ 0.5,1.0,2.0,4.0\}$ and generate 15 instances per parameter combination, yielding $1,500$ instances overall.

B. Computational Results

We first present our baseline results and then study the performance of vanilla rounding and heuristical rounding.

a) Baseline $\mathrm{MIP}_{\mathrm{MCF}}$*:* To obtain a near-optimal baseline solution for each of the 1,500 instances, we employ Gurobi 7.5.1 to solve the Mixed-Integer Programming Formulation 1 (using a single thread). We terminate the computation after 3 hours or when the *objective gap* falls below 1%, i.e. when the constructed solution is *provably* less than 1% off the optimum. On average the runtime per instance is 129.8 minutes [17].

Figure 3 gives an overview on these baseline solutions. In particular, based on the a priori profit computation, the number of requests which can be feasibly embedded is shown together with the acceptance ratio which on average lies around 75%. The rightmost plot depicts the objective gap, i.e. the quality guarantee proven by Gurobi, which is (on average) 6.8%.

b) Solving LP Formulation 2: To apply the rounding algorithms presented in Section IV-D, our novel LP Formulation 2 needs to be solved. Again, we employ Gurobi 7.5.1, specifically its Barrier algorithm with crossover. Figure 4 depicts the averaged runtime to solve the LP as well as to construct the LP. The latter is not negligible as the formulation contains up to 1,000k variables for some instances. The runtime increases from around 2 minutes for $|\mathcal{R}| = 40$ to around 7 minutes for $|\mathcal{R}| = 100$. The maximally observed runtime in our experiments amounted to roughly 18 minutes.

c) Vanilla Rounding $\mathrm{RR}_{\mathrm{MinLoad}}$*:* We first consider the performance of vanilla rounding. Concretely, we report on the best solution found within 1,000 rounding iterations, i.e. the solution minimizing resource augmentations and breaking ties among these by returning the solution of highest profit. Figure 5 (left) depicts the results. As can be seen, the algorithm achieves a profit between 50% and 140% compared to the best solution constructed by the MIP, while exceeding resource capacities mostly by 25% to 125% of the resource's capacity. The edge resource factor has a distinctive impact: for $\mathrm{ERF} = 4.0$ maximal resource loads mostly lie below 75%.

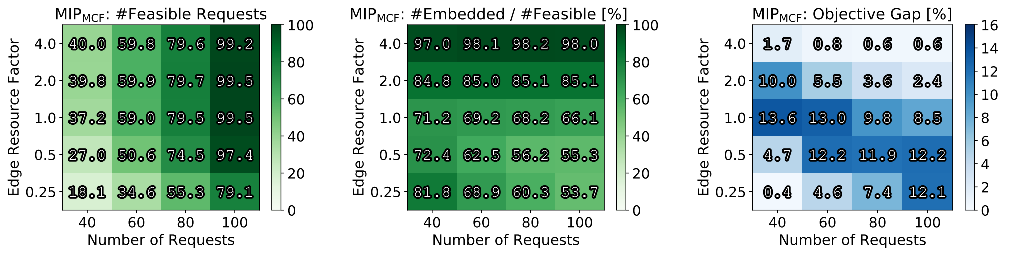

Fig. 3. Overview on baseline results computed using the MIP Formulation 1. Each cell averages the results over 75 instances. The feasibility of requests is obtained from (cost-optimally) embedding the requests to compute the profit a priori. The center plots depicts the acceptance ratio restricted to the feasible requests. The solution's quality is depicted on the right: the gap heavily depends on the edge resource factor but is on average less than 7%.

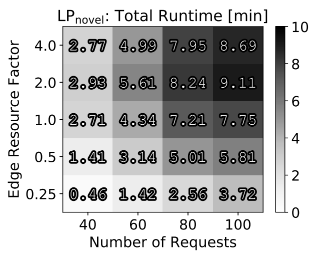

Fig. 4. Solution time of the novel LP Formulation 2 (including the construction time for the LP) using the Barrier algorithm of Gurobi 7.5.1.

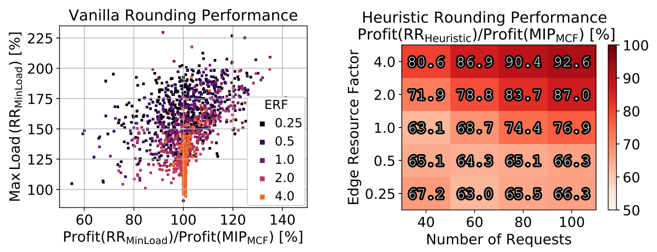

Fig. 5. Overview on results obtained using vanilla and heuristical rounding. Left: Solutions obtained via *vanilla* rounding minimizing the load. Each point corresponds to a single instance and is colored according to the instance's edge resource factor. 7 of 1,500 results lie outside the depicted area.
Right: The averaged profit of solutions obtained via heuristical rounding compared to the best baseline solution. Each cell averages 75 instances.

d) Heuristical Rounding $\mathrm{RR}_{\mathrm{Heuristic}}$: The results of the heuristical rounding are presented in Figure 5 (right). Again, 1,000 rounding iterations were considered. While for low edge resource factors, i.e. scarce edge resources, the solutions achieve around 65% of the profit of the MIP baseline, for larger edge resource factors, the relative performance exceeds 80%. Furthermore, the performance improves when increasing the number of requests. Overall, the average relative performance with respect to the baseline solutions is 73.8%, with the minimal one being 22.3%.

VI. Conclusion

This paper has initiated the study of approximation algorithms for the Virtual Network Embedding Problem supporting arbitrary substrate graphs and supporting virtual networks containing cycles. To obtain the approximation, we have derived a strong LP formulation for cactus request graphs. Our computational evaluation shows the practical significance of our work: obtained solutions achieve (on average) around 74% of the baseline's profit while *not augmenting capacities*.

We note that the developed approximation framework is independent of the how LP solutions are computed and decomposed. In particular, while the LP formulation presented in this paper is only applicable for cactus request graphs, our formulation can be generalized to arbitrary request graphs [18].

Acknowledgements

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

We thank Elias Döhne, Alexander Elvers, and Tom Koch for their significant contribution to our implementation [20].

References

[1] J. C. Mogul and L. Popa, "What we talk about when we talk about cloud network performance," *ACM SIGCOMM CCR*, vol. 42, no. 5, 2012.
[2] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. 3rd IEEE CloudNet*, October 2014.
[3] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 242–253.
[4] J. Napper, W. Haeffner, M. Stiemerling, D. R. Lopez, and J. Uttaro, "Service Function Chaining Use Cases in Mobile Networks," Internet-Draft, Apr. 2016. [Online]. Available: https://tools.ietf.org/html/draft-ietf-sfc-use-case-mobility-06
[5] A. Fischer, J. Botero, M. Till Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *Comm. Surveys Tutorials, IEEE*, vol. 15, no. 4, 2013.
[6] M. Rost and S. Schmid, "Charting the Complexity Landscape of Virtual Network Embeddings," in *Proceedings IFIP Networking*, 2018.
[7] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. IEEE INFOCOM*, 2009.
[8] R. Hartert et al., "A declarative and expressive approach to control forwarding paths in carrier-grade networks," in *SIGCOMM*, 2015.
[9] A. Jarray and A. Karmouch, "Decomposition approaches for virtual network embedding with one-shot node and link mapping," *IEEE/ACM Transactions on Networking*, vol. 23, no. 3, pp. 1012–1025, 2015.
[10] M. Rost, S. Schmid, and A. Feldmann, "It's About Time: On Optimal Virtual Network Embeddings under Temporal Flexibilities," in *Proc. IEEE IPDPS*, 2014, pp. 17–26.
[11] M. Rost, C. Fuerst, and S. Schmid, "Beyond the stars: Revisiting virtual cluster embeddings," in *Proc. ACM SIGCOMM Computer Communication Review (CCR)*, 2015.
[12] G. Even, M. Medina, and B. Patt-Shamir, "Online path computation and function placement in sdns," in *Proc. International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, 2016.
[13] G. Even, M. Rost, and S. Schmid, "An approximation algorithm for path computation and function placement in sdns," in *Proc. SIROCCO*, 2016.
[14] T. Lukovszki and S. Schmid, "Online admission control and embedding of service chains," in *Proc. 22nd SIROCCO*, 2015.
[15] N. Bansal, K.-W. Lee, V. Nagarajan, and M. Zafer, "Minimum congestion mapping in a cloud," in *Proc. ACM PODC*, 2011.
[16] M. Rost and S. Schmid, "Service chain and virtual network embeddings: Approximations using randomized rounding," Tech. Rep. arXiv:1604.02180 [cs.NI], April 2016.
[17] ——, "Virtual Network Embedding Approximations: Leveraging Randomized Rounding," Tech. Rep. arXiv:1803.03622 [cs.NI], March 2018.
[18] ——, "(FPT-)Approximation Algorithms for the Virtual Network Embedding Problem," Tech. Rep. arXiv:1803.04452 [cs.NI], March 2018.
[19] D. P. Dubhashi and A. Panconesi, *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press, 2009.
[20] E. Döhne, A. Elvers, T. Koch, and M. Rost, "Source code for the evaluation presented in this work," https://github.com/vnep-approx/evaluation-ifip-networking-2018.