Cost-Efficient Embedding of Virtual Networks With and Without Routing Flexibility

Balázs Németh[1] Yvonne-Anne Pignolet[2] Matthias Rost[3] Stefan Schmid[4] Balázs Vass[1]
[1] Budapest University of Technology and Economics, Hungary [2] DFINITY, Switzerland [3] TU Berlin, Germany
[4] Faculty of Computer Science, University of Vienna, Austria

***Abstract*—The Virtual Network Embedding Problem lies at the heart of many network resource allocation problems. Typically, this problem is studied in a variant where both the mapping of nodes (e.g., the virtual machines) as well as the routing is flexible and hence subject to optimization. However, many networks today are based on control planes with fixed paths, which cannot easily be changed. This paper presents the first approximation algorithm for the embedding cost of a set of virtual networks which do not have to provide routing flexibilities. In particular, we analytically prove that our algorithm achieves an XP approximation: a constant approximation in a resource augmentation model for scenarios with and without routing flexibilities. Our extensive simulations further indicate that our algorithm is not only of theoretical interest but is a practical solution.**

I. Introduction

Enabled by the increasing virtualization and programmability of communication technologies, networked systems are becoming increasingly flexible. This, in turn, introduces unprecedented optimization opportunities, e.g., in terms of resource allocations: network applications can be allocated where they are most useful and/or profitable. For example, frequently communicating virtual machines can be mapped close to each other in order to reduce communication overheads, or computationally intensive tasks can be offloaded to a nearby edge cloud to ensure minimal latency. The resource allocation flexibilities are only constrained by policy requirements and by the specific optimization objectives: e.g., the requirement to keep data geographically local or the desire to provide a predictable latency, or more generally, *performance*. In order to ensure such a predictable performance, flexibly mapped resources are often also interconnected by a network providing bandwidth guarantees.

Many of optimization problems underlying such flexible networks can essentially be formulated as a Virtual Network Embedding Problem (VNEP): The VNEP asks for a mapping of nodes and their interconnecting links from virtual networks to a substrate network under capacity constraints while optimizing a certain objective function, such as minimizing the resource costs (the widely studied *cost-minimization variant*). Interestingly, however, while there is a large body of literature on the VNEP, most existing algorithms either require a super-polynomial runtime or do not provide formal approximation guarantees. In general, the problem is known to be notoriously hard for many different variants [1].

ISBN 978-3-903176-28-7© 2020 IFIP

In this paper, in addition to the general VNEP, we are particularly interested in a specific version of the VNEP without routing flexibilities. Indeed, in many practical scenarios, the routing between two virtual node locations can be constrained. For example:

- ***Fixed control plane:*** In most networks today, from wide-area networks to datacenter networks, routes are determined by the network control plane autonomously. Most control planes use shortest path protocols, such as ECMP (equal-cost multi-path routing). For example, while virtual machines may be allocated flexibly in a datacenter, traffic between two VMs is restricted by ECMP.
- ***Real-time constraints:*** Also in many application domains requiring real-time guarantees, routing is restricted to shortest paths. For example, in industrial automation applications [2], from manufacturing, to mining and power generation, virtual networks often describe stream processing graphs interleaved with control algorithms. The nodes of these virtual networks are computational tasks (e.g., Fourier transforms or control functions) while the edges describe the processing order. By mapping frequently interacting computational tasks to topologically close substrate nodes and using shortest path routing, latency requirements can be met.

A. Contributions

We make two major technical contributions, the first one concerning the general VNEP problem and the second one concerning scenarios without routing flexibilities.

1) We present a polynomial-time cost approximation algorithm for the general VNEP. Our algorithm is based on randomized rounding, and to the best of our knowledge, is the first polynomial-time approximation algorithm with provable guarantees on the embedding cost.
2) We present a novel variant of the VNEP where routes cannot be optimized, e.g., due to performance considerations or technological constraints. We show how to adapt algorithms and analysis to this problem version, resulting in approximation guarantees again.

For both problems, we present a formal analysis and report on simulations, showing that our algorithm performs well both in theory and practice. We also find that restricting routing to shortest paths does not come at a high embedding price compared to fully flexible embeddings. We believe that our technical contribution, based on randomized rounding, as well as the model introduced in this paper, may be of independent interest beyond the specific problem considered in this paper.

As a contribution to the research community and to facilitate follow-up work, we publish our implementation and experimental data on GitHub to enable the reproducibility of our results[1].

B. Organization

The remainder of this paper is organized as follows. We introduce our formal model in Section II, and present the algorithmic background in Section III. Our approximations are given in Section IV, while its evaluation is presented in Section V. After discussing the related work in Section VI, we conclude our study in Section VII.

II. Model

Substrate Network. The *substrate network* is modeled as a weighted graph $G_S = (V_S, E_S, d_S)$, where the set of substrate nodes and substrate edges are denoted by V_S and E_S, respectively. There is an edge $e = (u,v) \in E_S$ exactly if there is a direct communication link between u and v. When referring to an arbitrary substrate element $x \in V_S \cup E_S$, we write $x \in G_S$. Each substrate element $x \in G_S$ has a limited capacity $d_S(x) \geq 0$. The capacity of a node $v \in V_S$ may represent the number of CPUs or memory constraints, while the capacity of an edge $(u,v) \in E_S$ refers to the available bandwidth. The number of nodes and edges of the substrate network is denoted by n_S and m_S, respectively. For simplicity, when referring to an arbitrary substrate element $x \in V_S \cup E_S$, we write $x \in G_S$.

Request Graphs. A *request graph* is modeled as a weighted directed graph $G_r = (V_r, E_r, d_r)$, where V_r and E_r denote the set of request nodes and edges or virtual nodes and edges, respectively. The request edges $E_r \subseteq V_r \times V_r$ represent the directed data flow among the request nodes. Each request element $x \in V_r \cup E_r$ has a demand $d_r(x) \geq 0$. Request nodes and edges are also referred to as virtual edges and nodes, respectively. For request node $i \in V_r$, the demand represents the resource requirements of i, e.g., the needed CPU capacity or memory.[2] In case of request edges, the demand represents the communication requirements of a pair of nodes (e.g., in terms of bandwidth). A set of requests $\mathcal{R}$ denotes a set of request graphs. For each request $r \in \mathcal{R}$, the number of nodes and edges are denoted by n_r and m_r, respectively.

Valid Mappings. Virtual nodes and edges can only be mapped on substrate nodes and edges of sufficient capacity. Furthermore, the customer or substrate provider may restrict the mapping of request nodes $i \in V_r$ and edges $(i,j) \in E_r$ by providing sets of *forbidden* substrate nodes $\overline{V}_S^i \subseteq V_S$ and edges $\overline{E}_S^{i,j} \subseteq E_S$. The set $\overline{V}_S^i$ may for example include substrate nodes too distant to the consumer or not suited to host the functionality of request node i. Similarly, the set $\overline{E}_S^{i,j}$ contains edges which must be avoided due to security or other technical policies. Accordingly, we denote the set of suitable substrate nodes for $i \in V_r$ by $V_S^{r,i} = \{u \in V_S \setminus \overline{V}_S^i | d_S(u) \geq d_r(i)\}$ and the set of suitable substrate edges for $(i,j) \in E_r$ by $E_S^{r,i,j} = \{(u,v) \in E_S \setminus \overline{E}_S^{i,j} | d_S(u,v) \geq d_r(i,j)\}$.

Furthermore, we denote by $d_{\max}(r,x)$ the maximal demand of any *single* request element of request $r \in \mathcal{R}$ on a single substrate resource $x \in G_S$. Specifically, the following holds:

$$d_{\max}(r,u) = \max\left(\{0\} \cup \left\{d_r(i) \mid i \in V_r : u \in V_S^{r,i}\}\right\}\right),$$
$$d_{\max}(r,u,v) = \max\left(\{0\} \cup \left\{d_r(i,j) \mid (i,j) \in E_r : (u,v) \in E_S^{r,i,j}\}\right\}\right).$$

Definition 1 (Valid Mapping). *A valid mapping of request $r \in \mathcal{R}$ to substrate G_S is a tuple (m_r^V, m_r^E) of functions $m_r^V : V_r \to V_S$ and $m_r^E : E_r \to \mathrm{P}(E_S)$, where $\mathrm{P}(E)$ denotes the set of simple substrate paths over the edge set E, such that:*

- *virtual nodes are placed on suitable substrate nodes, i.e. $m_r(v) \in V_S^{r,i}$, for all $i \in V_r$,*
- *the mapping $m_r^E(i,j)$ of edge $(i,j) \in E_r$ is a path connecting $m_r^V(i)$ and $m_r^V(j)$ which only uses allowed edges, i.e., $m_r^E \subseteq E_S^{r,i,j}$ holds.*

The set of all valid mappings of request r is denoted by $\mathcal{M}_r$.

Hence, a valid mapping enforces the validity of each single virtual element with respect to the set of forbidden nodes and edges. Cumulative resource allocations and their costs are defined as follows.

Definition 2 (Resource Allocation and Embedding Costs). *We denote by $A(m_r, x) \in \mathbb{R}_{\geq 0}$ the resource* allocation *induced by the valid mapping $m_r \in \mathcal{M}_r$ on substrate element $x \in G_S$. For $u \in V_S$ and $(u,v) \in E_S$, the following holds, respectively: $A(m_r, u) = \sum_{i \in V_r : m_r(i) = u} d_r(i)$ and $A(m_r, u, v) = \sum_{(i,j) \in E_r : (u,v) \in m_r^E(i,j)} d_r(i)$.*

Furthermore, the maximal allocation that a single virtual element may impose on a substrate resource $x \in G_S$ by $A_{max}(r,x) = \max_{m_r \in \mathcal{M}_r} A(m_r, x)$. Using $c_S(x)$ to denote the unit cost of substrate resource $x \in G_S$, the cost *$c_S(m_r)$ of a mapping $m_r \in \mathcal{M}_r$ is defined as $c_S(m_r) = \sum_{x \in G_S} c_S(x) \cdot A(m_r, x)$.*

An embedding that does not exceed the capacity of the substrate elements is called feasible.

Definition 3 (Feasible Embedding). *A set of mappings $\{m_r\}_{r \in \mathcal{R}'}$ over a set of requests $\mathcal{R}' \subseteq \mathcal{R}$ constitutes a* feasible embedding, *if and only if $\sum_{r \in \mathcal{R}'} A(m_r, x) \leq d_S(x)$ holds for every $x \in G_S$. A single mapping m_r is feasible, if this holds for the singleton set $\{m_r\}$.*

Problem Definitions. In the following, both the profit and cost variants of the Virtual Network Embedding Problem

[1]https://github.com/vnep-approx/evaluation-ifip-networking-2020

[2]Instead of a single value, the requirements can be represented by a vector with one entry for each of the different types of resources. For simplicity, we present the single value case in this paper, all our results can be extended to the more general case by normalization and vector norms.

(VNEP) and its relaxation, the Valid Mapping Problem (VMP), are introduced.

The first problem is only concerned with the validity of the mapping, ignoring capacity limitations:

Definition 4 (Cost Valid Mapping Problem (CVMP))**.** *Given a single request r and considering substrate costs, a solution to the* CVMP *decides whether there exists a valid mapping, and, if so, returns a valid mapping m_r minimizing the cost $c_S(m_r)$.*

The VNEP accounts for feasibility with regards to capacities, and comes in two flavors: profit and cost:

Definition 5 (Cost Virtual Network Embedding Problem (CVNEP))**.** *The Cost VNEP returns a minimal cost feasible embedding for the request set $\mathcal{R}$, if one exists.*

Definition 6 (Profit Virtual Network Embedding Problem (PVNEP))**.** *Given a profit $b_r \geq 0$ for each request $r \in \mathcal{R}$, the profit VNEP returns a feasible embedding $\{m_r\}_{r \in \mathcal{R}'}$ of a subset of requests $\mathcal{R}' \subseteq \mathcal{R}$ maximizing the profit $\sum_{r \in \mathcal{R}'} b_r$.*

Restricted Routing. While the VNEP poses a very general problem formulation, some applications require more restrictions. Hence, in addition to the general problem formulation, we are particularly interested in solving the Cost VNEP under additional routing restrictions. To this end, a modified problem version also specifies a routing $\mathcal{P}$ over G_S such that for each pair of substrate nodes $u, v \in V_S$ only a single predefined path $\mathcal{P}(u, v) \in \mathrm{P}(E_S)$ may be used to route flow from u to v. Note that this differs from the concept of the forbidden set of links for a request, and restricts the traffic between a source and a destination, often regardless of the flow.

Definition 7 (VNEP without Routing Flexibilities)**.** *Given a routing $\mathcal{P}$, a mapping $m_r = (m_r^V, m_r^E)$ is valid if each virtual edge $(i,j) \in E_r$ takes the predefined path, i.e., $m_r^E(i,j) = \mathcal{P}(m_r^V(i), m_r^V(j))$ holds. Under this adaptation, CVNEP and PVNEP return, a feasible embedding minimizing the cost or maximizing the profit for a request set $\mathcal{R}$ as input.*

An algorithm produces an (α, β, γ)*-approximation* of the CVNEP (or PVNEP) if any solution features at most $\alpha > 1$ times the cost (or at least $1/\alpha$ times the profit) of an optimal solution, and the allocations on nodes and edges are within factors of $\beta \geq 1$ and $\gamma \geq 1$ of the original capacities respectively, with high probability (whp).

III. Preliminaries

In this paper, we will make use of the following graph-theoretical concepts.

Definition 8 (Tree decomposition and Treewidth [3])**.** *Given a graph $G_r = (V_r, E_r)$, a tree decomposition of G_r is a pair $\mathcal{T}_r = (T_r, \mathcal{B}_r)$, consisting of an undirected tree $T_r = (V_T, E_T)$ and a family of node bags $\mathcal{B}_r = \{B_t\}_{t \in V_T}$ with $B_t \subseteq V_r$, for which the following hold:*

1) *for node $i \in V_r$, the set $V_T^{-1}(i) = \{t \in V_T | i \in B_t\}$ of tree nodes containing i is connected in T_r, and*
2) *each node and each edge is contained in at least one bag.*

A tree decomposition is small *if $B_{t_1} \not\subseteq B_{t_2}$ holds for all $t_1, t_2, \in V_T$ with $t_1 \neq t_2$. The treewidth $tw(\mathcal{T}_r)$ of a tree decomposition $\mathcal{T}_r$ is the maximum bag size minus 1, i.e. $tw(\mathcal{T}_r) = \max_{t \in V_T} |B_t| - 1$. The treewidth of a graph, $tw(G_r)$ is the minimum widths of all its tree decompositions.*

While computing a tree decomposition of minimal treewidth is itself a NP-hard problem, it is fixed-parameter tractable (FPT), i.e., it can be computed in polynomial time when the treewidth is bounded by a constant [4]. Furthermore, any tree decomposition can be transformed into a small one in linear time [4]. As discussed in [5], many well-known graph classes, e.g., series-parallel graphs and outerplanar graphs, have bounded treewidth.

We next review the algorithmic framework that enabled obtaining the first approximations for the profit variant of the VNEP [6]. In this paper, we will build upon these techniques to obtain an approximation of the cost variant of VNEP, both in a model with and a model without routing flexibilities.

Using tree decompositions, Rost et al. showed in [7] that the Cost Valid Mapping Problem (CVMP, cf. Section II) can be solved optimally using Dynamic Programming on tree decompositions with time complexity exponential in the tree decomposition's treewidth. The DynVMP algorithm harnesses the following observation: for request graphs of limited size, optimal valid mappings can readily be computed by *enumerating* all potential node mappings for each node bag and adding the respective lowest valid paths costs to the respective node mapping costs. Given this insight, first, for all leaves of a tree decomposition $\mathcal{T}_r$ cost-optimal valid mappings are computed (if a valid mapping exists). Then, for each inner node $t_I \in T_V$, the (partial) cost optimal mappings of the induced graph $G[B_{t_I}]$ of node bag B_{t_I} are computed for each node mapping of B_{t_I} by adding the respective costs of the optimal mappings of the children. For more in-depth information on the DynVMP algorithm, we refer the reader to [7].

Note that, the runtime complexity of DynVMP depends on the maximal treewidth of any of the requests, more precisely it is bounded by $\mathcal{O}({n_r}^3 \cdot {n_S}^{2 \cdot \mathrm{tw}(\mathcal{T}_r)+3})$. Hence, when all requests have bounded treewidth, as is the case for example for cactus or outerplanar graphs, then the runtime is polynomial.

To obtain an approximation of the profit VNEP, Rost and Schmid proposed in [6] to apply randomized rounding after preprocessing all requests (cf. Algorithm 4). Specifically, first, all requests are removed that cannot be fully embedded in the absence of other requests and then each mapping m_r^k is chosen with probability f_r^k for each request $r \in \mathcal{R}$. This process is iterated until an approximate solution is obtained or the maximal number of rounding tries is exceeded. In this paper, we show how to build upon these results to devise an approximation algorithm for the cost variant of the embedding problem. Note that we cannot use the same approach unchanged, as the approximation guarantee in the profit variant relies on the fact that we can discard any request

LP Formulation 1: Enumerative Cost VNEP – Primal

$$\min \sum_{r\in\mathcal{R}, m_r^k\in\mathcal{M}_r} f_r^k \cdot c_S(m_r^k) \qquad (1)$$

$$\sum_{m_r^r\in\mathcal{M}_r} f_r^k = 1 \qquad \forall r\in\mathcal{R} \qquad (2)$$

$$\sum_{r\in\mathcal{R}, m_r\in\mathcal{M}_r} f_r^k A(m_r^k, x) \leq d_S(x) \qquad \forall x\in G_S \qquad (3)$$

$$f_r^k \geq 0 \qquad r\in\mathcal{R}, \forall m_r^k\in\mathcal{M}_r \qquad (4)$$

LP Formulation 2: Enumerative Cost VNEP – Dual

$$\max \sum_{r\in\mathcal{R}} \lambda_r + \sum_{x\in G_S} \mu_x \cdot d_S(x) \qquad (5)$$

$$\lambda_r \in \mathbb{R} \qquad \forall r\in\mathcal{R} \qquad (6)$$

$$\mu_x \leq 0 \qquad \forall x\in G_S \qquad (7)$$

$$\lambda_r + \sum_{x\in G_S} \mu_x \cdot A(m_r^k, x) \leq c_S(m_r^k) \qquad \forall r\in\mathcal{R}, m_r^k\in\mathcal{M}_r \qquad (8)$$

and still have a valid solution, though maybe less profitable solution. In other words, in addition to the degrees of freedom stemming from the embedding options, we have the choice of the subset of requests that actually are to be embedded without violating the capacity bounds too much. In contrast, the cost variant of the problem can only play with the mapping of the requests and must ensure that all of them are embedded.

IV. Approximations for Cost VNEP

In this section, the first approximation for the cost variant of the VNEP is given, yielding solutions for the problem variants with and without routing flexibilities.

A. General Approximation

While relying on the algorithmic preliminaries presented above, to obtain the approximation result, several obstacles need to be overcome. Firstly, while for the profit VNEP a feasible solution can always be easily constructed by simply not embedding any request, finding a feasible solution to the cost problem is itself NP-complete [1]. Furthermore, an approach purely relying on the randomized rounding of fractional Linear Programming solutions, can in general not guarantee any approximation guarantee on the achieved cost. Accordingly, we first discuss how to compute optimal Linear Programming solutions for the Cost VNEP (cf. LP Formulation 1) and then present the novel approximation algorithm having a *deterministic* approximation guarantee on the achieved cost.

Computing LP Solutions for the Cost VNEP. To compute fractional solutions for the Cost VNEP, we introduce LP Formulation 1 (later we will also need the dual LP Formulation 2). Using the approach to enumerate all valid mappings, Constraint 2 enforces that each request needs to be fully embedded, while this fractional solution must be feasible with respect to the capacities (cf. Constraint 3). The objective function clearly captures the cost of the fractional solution by multiplying each mapping cost with the corresponding *weight* variables. To solve this LP Formulation, we present a novel column generation approach as Algorithm 3. The algorithm first computes a feasible fractional embedding (disregarding the cost) by employing the algorithm for the profit LP Formulation in [6]. Given that the fractional profit LP can be solved optimally, the returned solution either embeds each request fully or returns an incomplete solution. In the latter case, there cannot exist a feasible embedding for the whole request set and $\perp$ is returned. The remainder of the algorithm does not use the normal cost function but the cost function $c_\mu(x),\ x\in G_S$ which is based on *dual* variables μ_x.

Theorem 1. *Using Algorithm 3, LP Formulation 1 can be solved optimally using the* DynVMP *algorithm in time* $\mathcal{O}\left(\mathsf{poly}\left(\sum_{r\in\mathcal{R}} n_r^3 \cdot n_S^{2\cdot tw(\mathcal{T}_r)+3}\right)\right)$ *given specific tree decompositions* $\mathcal{T}_r$ *for each request* $r\in\mathcal{R}$*, if a feasible embedding exists.*

Proof: As argued above, the output of the profit LP indicates whether a feasible fractional embedding exists and only if such a solution exists, one is returned (otherwise the algorithm returns $\perp$). Hence, initializing the sets of valid mappings $\hat{\mathcal{M}}_r$ according to the mappings used by the profit LP, also a feasible LP solution for the cost LP (cf. LP Form. 1) must exist. Hence, each LP computation will yield a primal feasible solution together with corresponding dual variables.

To see that Algorithm 3 returns an optimal cost solution, consider now the column generation of Lines 5 to 9. Noting

Algorithm 3: Solving LP 1 via Column Generation

```
   Output: optimal cost LP solution or ⊥ if none exists
 1 compute profit LP solution [6] with $b_r = 1$ for each $r\in\mathcal{R}$
 2 if LP's solution profit less than $|\mathcal{R}|$ then
 3     return ⊥ as no solution can exist
 4 initialize sets of valid mappings $\hat{\mathcal{M}}_r$ using the above LP
     solution
 5 do
 6     compute solution to LP 1 over mapping sets $\{\hat{\mathcal{M}}_r\}_{r\in\mathcal{R}}$
 7     foreach $r\in\mathcal{R}$ do
 8         compute minimal cost mapping $\hat{m}_r$ using DynVMP
             under costs $c_\mu(x) := c_S(x) - \mu_x$ for $x\in G_S$
 9         if $c_\mu(\hat{m}_r) < \lambda_r$ then add $\hat{m}_r$ to $\hat{\mathcal{M}}_r$
10 while any mapping violating Constraint 8 was added
11 return last computed primal LP solution
```

Algorithm 4: Approximation of Profit VNEP [6]

```
   Input  : substrate $G_S$, requests $\mathcal{R}$ with profit $b_r$ for each
            $r\in\mathcal{R}$, approx. factors $\alpha,\beta,\gamma$ and $N$ rounding tries
   Output: $(\alpha,\beta,\gamma)$-approximate solution with high probability
 1 foreach $r\in\mathcal{R}$ do              // preprocess requests
 2     compute profit LP solution for request $r$
 3     remove $r$ from $\mathcal{R}$ if solution's profit is less than $b_r$
 4 compute profit LP solution for all requests $\mathcal{R}$
 5 do                          // perform randomized rounding
 6     foreach $r\in\mathcal{R}$ embed $r$ using $m_r^k$ with probability $f_r^k$
       // $r$ is rejected with prob. $1-\sum_k f_r^k$
 7 while solution not $(\alpha,\beta,\gamma)$-approximate and $\leq N$ tries
```

the adapted definition of c_μ in Line 8, we first show that Constraint 8 can be equivalently stated by $c_\mu(m_r^k) \geq \lambda_r$ for each valid mapping m_r^k and each request $r \in \mathscr{R}$.

$$\lambda_r + \sum_{x \in G_S} \mu_x \cdot A(m_r^k, x) \leq c_S(m_r^k) \quad (9)$$

$$\Leftrightarrow \quad -\sum_{x \in G_S} c_S(x) \cdot A(m_r^k, x) + \sum_{x \in G_S} \mu_x \cdot A(m_r^k, x) \leq -\lambda_r \quad (10)$$

$$\Leftrightarrow \quad \sum_{x \in G_S} (c_S(x) - \mu_x) \cdot A(m_r^k, x) \geq \lambda_r \quad (11)$$

$$\Leftrightarrow \quad c_\mu(m_r^k) \geq \lambda_r \quad (12)$$

Above, in Equations 10 and 12, the definition of the costs of mappings (see Definition 2) was used to first decompose and then aggregate the respective costs again. Given this equivalence of the constraints to $c_\mu(m_r^k) \geq \lambda_r$, the column generation approach only adds valid mappings in Line 9 for which the dual constraints are violated. Furthermore, as $\mu_x \leq 0$ holds for all resources $x \in G_S$ (cf. Constraint 7), the cost function c_μ is positive for all resources $x \in G_S$, which is required for DynVMP to work.

Lastly, considering the runtime, we note that this is due to the result by Grötschel, Lovász and Schrijver [8] and the runtime of DynVMP. ■

Description and Analysis of the Cost Approximation. Given the ability to also solve the fractional Cost VNEP according to LP Formulation 1, we now present the first VNEP cost approximation based on randomized rounding as Algorithm 5. The approximation scheme differs from the one for the profit (cf Algorithm 4) by performing an important post-processing step to ensure a deterministic approximation guarantee for the cost: if a feasible fractional solution exists, in Lines 4 to 7 all *costly* mappings are pruned by setting their respective weight variable f_r^k to 0. Specifically, considering an approximation guarantee on the cost of $\alpha > 1$, first the weighted (average) cost WC_r is computed and then all mappings whose cost is larger than $\alpha \cdot WC_r$ are pruned. Accordingly, the following deterministic approximation guarantee is easy to establish:

Lemma 9. *Algorithm 5 only returns α-approximate solutions.*

Proof: Let c_{frac} denote the cost of the LP solution computed in Line 1 and let c_{opt} denote the minimum cost of any feasible integral embedding. Clearly, $c_{frac} \leq c_{opt}$ holds, as any integral solution of LP Formulation 1, i.e., $f_r^k \in \{0,1\}$ must hold, is also a fractional solution and Algorithm 3 returns the minimum cost fractional solution. Furthermore, $c_{frac} = \sum_{r \in \mathscr{R}} WC_r$ holds and as all mappings of cost larger than $\alpha \cdot WC_r$ are removed for each request, the returned solution's cost is upper bounded by $\alpha \cdot \sum_{r \in \mathscr{R}} WC_r = \alpha \cdot c_{frac} \leq \alpha \cdot c_{opt}$. ■

After having removed the costly mappings, the weights of the remaining mappings are normalized. Specifically, the weights of the remaining mappings are scaled by $1/\sum_{m_r^k \in \hat{\mathcal{M}}_r} f_r^k$, such that $\sum_{m_r^k \in \hat{\mathcal{M}}_r} f_r^k = 1$ holds again. Importantly, by pruning mappings and scaling the weights afterward, the resulting *fractional* solution, which is then rounded, will exceed capacities by a certain amount. Accordingly, we analyze the maximal scaling factor in dependence of α.

Algorithm 5: Approximation of Cost VNEP

```
Input  : substrate $G_S$ with cost $c_S : G_S \to \mathbb{R}_{\geq 0}$, requests $\mathscr{R}$,
         approx. factors $\alpha, \beta, \gamma$ and $N$ rounding tries
Output: $(\alpha, \beta, \gamma)$-approx. solution whp. or $\perp$ if none exists
1  compute optimal fractional cost LP solution using Algorithm 3
2  if LP solution is $\perp$ then
3      return $\perp$     // no feasible embedding exists
   // post-process: prune costly mappings
4  foreach $r \in \mathscr{R}$ do
5      compute $WC_r \leftarrow \sum_{m_r^k \in \hat{\mathcal{M}}_r} f_r^k \cdot c_S(m_r^k)$
6      set $f_r^k \leftarrow 0$ for $m_r^k \in \mathcal{M}_r$ with $c_S(m_r^k) > \alpha \cdot WC_r$
7      normalize weights such that $\sum_{m_r^k \in \hat{\mathcal{M}}_r} f_r^k = 1$ holds again
8  do              // perform randomized rounding
9      foreach $r \in \mathscr{R}$ embed $r$ using $m_r^k$ with probability $f_r^k$
10 while solution not $(\alpha, \beta, \gamma)$-approximate and $\leq N$ tries;
```

Lemma 10. *The weights of the remaining mappings are scaled by at most a factor $\alpha/(\alpha - 1)$ in Line 7 of Algorithm 3.*

Proof: Let $\sigma_r = \sum_{m_r^k \in \hat{\mathcal{M}}_r : c_S(m_r^k) \leq \alpha \cdot WC_r} f_r^k$ denote the sum of the weights of the mappings of cost bounded by $\alpha \cdot WC_r$. For the sake of contradiction, assume that $\sigma_r < 1 - 1/\alpha$ holds for any request $r \in \mathscr{R}$. By the definition of WC_r and the assumption on σ_r, we obtain the following contradiction:

$$WC_r = \sum_{m_r^k \in \hat{\mathcal{M}}_r} f_r^k \cdot c_S(m_r^k) \quad (13)$$

$$\geq \sum_{m_r^k \in \hat{\mathcal{M}}_r : c_S(m_r^k) > \alpha \cdot WC_r} f_r^k \cdot c_S(m_r^k) \quad (14)$$

$$\geq \sum_{m_r^k \in \hat{\mathcal{M}}_r : c_S(m_r^k) > \alpha \cdot WC_r} f_r^k \cdot \alpha \cdot WC_r \quad (15)$$

$$\geq (1 - \sigma_r) \cdot \alpha \cdot WC_r > WC_r \quad (16)$$

The Inequality 14 holds as only a subset of the requests is considered and Inequality 15 follows as all the considered decompositions have a cost of at least $\alpha \cdot WC_r$. The first inequality of Equation 16 then follows as each request is fully embedded and the cumulative weight of the costly mappings is bounded by $(1 - \sigma_r)$ by assumption. As Equation 16 yields a contradiction, $\sigma_r \geq 1 - 1/\alpha$ must hold. Accordingly, after pruning the costly mappings, the remaining mappings are scaled by at most a factor of $1/(1 - 1/\alpha) = \alpha/(\alpha - 1)$. ■

Accordingly, the fractional solution might violate capacities by a factor of $\alpha/(\alpha - 1)$ and the following holds:

$$\sum_{r \in \mathscr{R}} \sum_{m_r^k \in \hat{\mathcal{M}}_r} f_r^k \cdot A(m_r^k, x) \leq \alpha/(\alpha - 1) \cdot d_S(x). \quad (17)$$

To prove bounds on node and edge capacity violations, i.e., β and γ, we employ the following Lemma of Rost and Schmid [6], which was proven for the profit approximation. However, as the randomized rounding procedure for the profit and the cost variant is identical, this lemma also pertains to our new cost approximation.

Lemma 11 ([6])**.** *Denote by A_x the random variable describing the allocations on resource $x \in G_S$ by performing randomized rounding. Let $0 < \varepsilon \leq 1$ be chosen such that $d_{\max}(r,x)/d_S(x) \leq \varepsilon$ holds for $r \in \mathcal{R}$, then*

$$\mathbb{P}(A_x \geq \mathbb{E}(A_x) + \delta(\kappa) \cdot d_S(x)) \leq \kappa^{-4} \tag{18}$$

holds for $\delta(\kappa) = \varepsilon \cdot \sqrt{2 \cdot \Delta(x) \cdot \log(\kappa)}$ with $\kappa > 2$ and $\Delta(x) = \sum_{r \in \mathcal{R}: d_{\max}(r,x)>0} (A_{\max}(r,x)/d_{\max}(r,x))^2$.

Alas, given the above results, the following main approximation result is obtained.

Theorem 2 (Approximation for the Cost VNEP)**.** *Algorithm 5 returns an (α,β,γ)-approximate solution for CVNEP, where $\beta,\gamma \geq 1$ are defined as*

$$\beta = \alpha/(\alpha-1) + \varepsilon \cdot \sqrt{2 \cdot \Delta(V_S) \cdot \log(n_S)}$$
$$\gamma = \alpha/(\alpha-1) + \varepsilon \cdot \sqrt{2 \cdot \Delta(E_S) \cdot \log(m_S)}$$

with $\Delta(X) = \max_{x \in X} \sum_{r \in \mathcal{R}: d_{\max}(r,x)>0} (A_{\max}(r,x)/d_{\max}(r,x))^2$ being the maximal sum of squared maximal allocation-to-capacity ratios over the resource set X and the maximum demand-to-capacity ratio $\varepsilon = \max_{r \in \mathcal{R}, x \in G_S} d_{\max}(r,x)/d_S(x)$. The runtime lies in $\mathcal{O}\left(\text{poly}\left(\sum_{r \in \mathcal{R}} n_r^3 \cdot n_S^{2 \cdot tw(\mathcal{T}_r)+3}\right)\right)$.

Proof: The *deterministic* approximation guarantee of α follows from Lemma 9. Considering the resource augmentation factors β and γ, we first note that the expected allocation $\mathbb{E}(A_x)$ on resource $x \in G_S$ is bounded by $\alpha/(\alpha-1)$ times the resource's capacity by Equation 17. Hence, applying Lemma 11 using $\kappa = n_S$, the probability to exceed the capacity of any node resource by a factor of the above-defined value β is bounded by ${n_S}^{-4}$. Analogously, the probability to exceed the capacity of any edge resource by a factor of γ is bounded by ${m_s}^{-4}$. As there are only n_S nodes and at most n_S^2 edges, the probability to violate *any* resource by factors of β or γ, respectively, is bounded by $1/n_S^3 + 1/n_S^2$, which is less than $3/8$ for $n_S \geq 2$. Hence, the probability to obtain a (α,β,γ)-approximate solution within N rounding iterations is lower bounded by $1-(3/8)^N$, i.e., approximate solutions are returned *with high probability*. Lastly, the runtime of the approximation is upper bounded by the time to construct the fractional LP solution and the result follows. ■

Notably, and in contrast to the profit approximation, our approximation allows to trade off the approximation guarantee for the cost with the resource augmentations. Specifically, compared to the profit approximation, the resource augmentations increase by the *additive* term $\alpha/(\alpha-1)-1 > 0$. Hence, choosing, for example, $\alpha = 2$, the resource augmentations are only increased by one compared to the profit approximation. Lastly, we note that Algorithm 5 is an XP-approximation, as its runtime is exponential in the request graphs' treewidth. However, this is the best one can hope for, as, e.g., the VNEP is inapproximable for general graphs (unless $P = NP$).

B. Approximation without Routing Flexibilities

The model without routing flexibilities is a restriction of the Cost VNEP, such that flows between substrate nodes must be routed along predefined paths and location-bound request nodes must be assigned to the correct substate nodes. In the following, we show that the DYNVMP algorithm can be adapted to only use these predefined paths. Hence, given the ability to compute such restricted fractional VNEP solutions efficiently, the cost approximation for the VNEP carries over to the variant without routing flexibilities.

As discussed in Section III, the DYNVMP algorithm works by computing minimum cost valid mappings for all potential node mappings of subgraphs of the given request. Specifically, for each *node bag* of the respective tree decomposition all (valid) node mappings are enumerated. Computing the minimum costs under fixed node mappings reduces to summing up the respective node mapping costs and computing shortest valid paths between the fixed endpoints for each contained virtual edge. Hence, under a fixed routing scheme, computing shortest paths is superfluous and the costs for edge mappings are just a function of the routing scheme.

Hence, adapting the DYNVMP accordingly, only valid mappings restricted on the given routing are obtained. This way, Algorithm 3 can be readily applied to compute optimal LP solutions *for the model without routing flexibilities*. As our presented approximation for the Cost VNEP does *only* depend on the ability to compute optimal LP solutions, we have the following corollary.

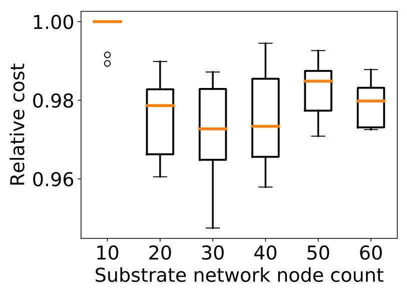
Corollary 12 (Cost Approximation without Routing Flexibilities)**.** *Adapting* DYNVMP *to only consider predefined routes, Algorithm 5 yields an approximation for the variant without routing flexibilities, of the same quality guarantees as stated in Theorem 2.*

V. EVALUATIONS

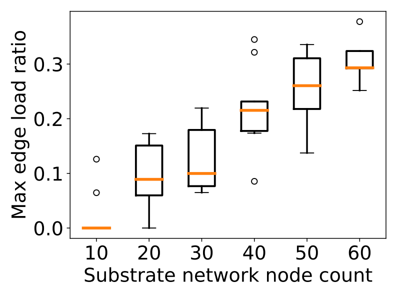
In this section we evaluate the performance of Algorithm 5. We study the effect of fixed path routing by comparing simulation results to solve the Cost VNEP with and without routing flexibility. Furthermore, we analyze our algorithm in terms of cost approximation, running time and resource capacity violation ratios by increasing input size on synthetic and real-world topologies.

Synthetic substrate graph topology. Industrial communication network topologies are often similar to the cactus graph class as tree-like hierarchical structures mix with cycles for redundancy [2]. Cacti are undirected graphs, where each node is contained by at most one cycle and thus they fit the above description. Furthermore, it has been shown that almost a third of the topologies in well-known internet topology-datasets are cactus graphs, too [9]. We generate random cactus graphs for our substrate network structure having a similar number of nodes in fix-sized cycles and trees. Substrate resource capacities and costs on both edges and nodes are set to 1.0.

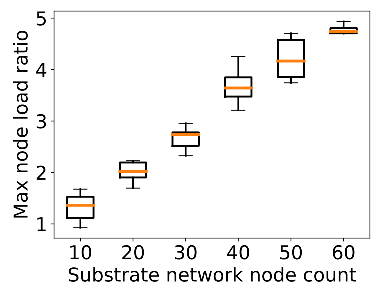
Real-world substrate graph topology. In addition to random graphs, we take the topology of three real-world networks as host graphs to prove the generality and applicability of our results. We use *Geant2012*, *GtsHungary* and *SwitchL3* from the Internet Topology Zoo [10] which have 40, 30 and



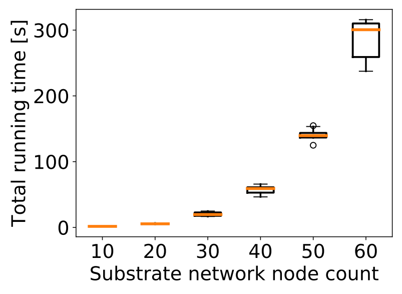
(a) Cost of integer solution relative to LP bound



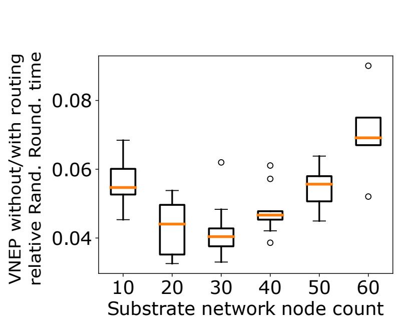
(b) Maximum edge load ratio



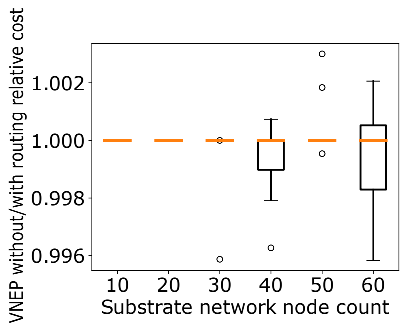
(c) Maximum node load ratio



(d) Total runtime to solve the version without flexibility



(e) Relative rounding time: arbitrary/fixed path



(f) Relative cost: arbitrary/fixed path

Fig. 1: Experimental results of the Cost VNEP (with and without flexibility) approximation on synthetic substrates.

42 nodes, representing a Europe-wide, Hungarian and Swiss country-wide networks respectively. Capacity and cost settings are set to 1.0 for all graph elements as in the synthetic case.

Request graph topology. We use series-parallel graphs (SPGs, having treewidth at most two [5]) to mimic request graph structures resulting from sense-process-actuate control loops, map-reduce applications, and path-like service chains. We generate SPG requests by randomly alternating between the SPG generating operations until we reach the desired node count. We remove any parallel or loop edges remaining in the generated SPG to conform with our request graph definition. Scenarios with $|\mathscr{R}| = 5$ are considered, where the sum of the request node numbers add up to twice the node count of the substrate, i.e.: $2n_S = \sum_{r\in\mathscr{R}} n_r$.

Resource demands are randomly generated for each request node and edge, controlled by aggregate parameters as in earlier evaluations of the profit variant [11]. The sum of the generated node resource demands adds up to 40% of the total resource capacities of the substrate network nodes. Link resource scarcity is set to have 100% utilization on all substrate links if all request links would be mapped to paths consisting of exactly 10 substrate links. Location bound request nodes play an important role in many application scenarios, as often only certain substrate nodes offer a specific function due to hardware, regulatory, or configuration constraints. Thus, we randomly select 10% of all request nodes in $\mathscr{R}$ to bound them to a randomly picked substrate node with sufficient capacity. This request graph generation method is used for both substrate graph scenarios.

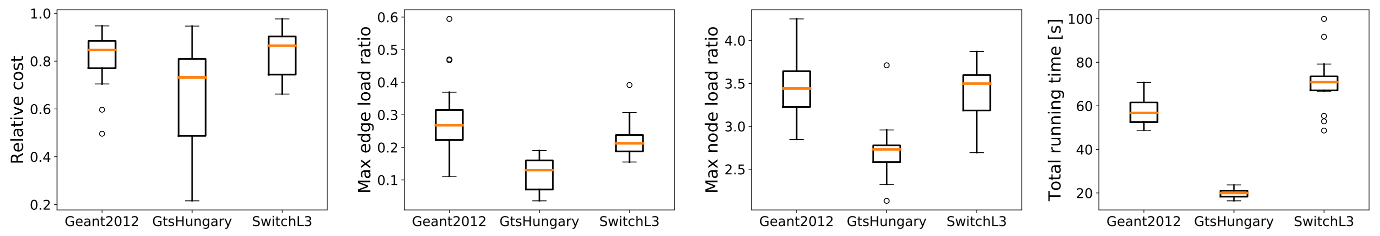
Results. We executed experiments on various instance sizes and topologies, varying the number of request nodes accordingly to keep the $2n_S = \sum_{r\in\mathscr{R}} n_r$ ratio. The experiments were executed on an OpenStack virtual machine with 32GB RAM and 4 vCPU-s, using Gurobi 7.5.2[3] to solve the LPs.

Each scenario was executed 10 times for the synthetic substrates and 20 times for the real-world topologies with different randomization seeds. During the evaluations a scenario is considered feasible if Algorithm 5 finds an $(\alpha = 2, \beta = 5, \gamma = 2)$-approximate integer solution. The generated scenarios are predominantly feasible with few exceptions. Fig. 1 and Fig. 2 summarize our findings.

The synthetic cases are shown in Fig. 1. The integer solutions generated by Algorithm 5 cost approximately the same as the best solutions of the LP relaxation by Algorithm 3: the relative cost of the integer solution, i.e. its cost divided by the LP relaxation's cost was between 0.95 and 1, indifferently of the substrate network size (see Fig. 1a). This is possible since while LP solutions are always *feasible*, outputs of Algorithm 5 are only (α,β,γ)-*approximate*, and violating resource constraints (in our experience, especially node constraints) allows choosing more compact (thus cheaper) valid mappings of requests. Note that the cheapest non-violating integer solution would be more expensive than the LP bound, and thus more expensive than our solution.

Fig. 1b-1c shows that while substrate edges are underused, node resources are violated consistently. As we can see, the

[3]Commercial MILP solver, https://www.gurobi.com/



(a) Cost of integer solution / LP bound (b) Maximum edge load ratio (c) Maximum node load ratio (d) Total runtime to solve Cost VNEP

Fig. 2: Experimental results of the Cost VNEP on Internet Topology Zoo instances as substrate topologies.

load ratios are growing with the size of the substrate network. This is due to the experimental setting and the specialty of DynVMP: while DynVMP tends to collocate the request nodes, bigger substrate networks have to handle bigger request graphs. Node capacity violations in Fig. 1c for lower instance sizes are below the specified $\beta = 5$, while for higher instances, the approximation gets tighter and the resource augmenting integer feasibility slightly decreases.

Fig. 1d shows the total running time of our algorithm containing all initialization steps, fractional solution calculation and rounding time for solving the problem version without routing flexibility. The running time is dominated by the preprocessing steps required for building the data structures of DynVMP, secondly the execution of Algorithm 3, which widely depends on the quality of the initializing mappings.

We simulated each scenario for the case when routing paths are given, i.e. solving the problem without flexibility, with precomputed shortest paths, and when routing paths are left for the optimizer to determine, i.e. solving the Cost VNEP. Fig. 1e shows the ratio of the execution times of randomized rounding in Algorithm 5 with arbitrary routing divided by the setting when fixed routing paths are given. We see that rounding solutions with routing flexibility takes 4-8% of the rounding time of the fixed routing case. We observed that for the version without flexibility the number of generated valid mappings (cf. Algorithm 3) increased significantly. This may be explained by the stringent routing restrictions requiring many more additional mapping configurations to use the resources of least cost homogeneously.

Concluding the analysis of the synthetic substrate experiments, Fig. 1f compares the achieved integer costs in the two routing settings. We observe that the total costs of the cases barely differ (less than 1%), so the additional constraints of routing restrictions do not incur significant changes in terms of costs. In general, the restriction to shortest path routing (precalculated for the version without flexibility) does not come at a high price in terms of embedding: the solution is close to optimal.

Results on the real-world network topologies are shown in Fig. 2. Request graph parameters are identical to the previous experiments to provide a reasonable comparison. As shown by Fig. 2a, solution costs relative to the LP bound are lower than in synthetic cases. Instead of at most 0.95, the solution qualities are around 0.85-0.7, meaning even cheaper solutions. Fig. 2b and 2c depict the edge and node capacity violation ratios, showing very similar results to the synthetic cases. At the node counts of *Geant2012* and *SwitchL3* (40 and 42 respectively) capacity violations are around 0.2 and 3.5 for the edge and node load ratios respectivly, as observed in the synthetic cases (cf. Fig. 1b and 1c). The same comparison hold for the *GtsHungary* topology. In conclusion, solution qualities are slightly better, but no significant difference can be observed for the capacity violations on the real-world topologies. Similar to the total runtimes of the version without flexibility on the synthetic cases with instance sizes 30-40, the total runtimes of the Cost VNEP on the topology zoo instances are shown in Fig. 2d.

The simulation results demonstrate our cost approximation algorithm's practical applicability both with and without fixed path routing constraints for the notoriously difficult VNEP.

VI. Related Work

Different flavors of problems and solutions for the allocation of computation and storage requests in cloud, fog, and edge computing scenarios are emerging, taking different constraints and objectives into account. In the last IEEE Infocom conference alone, more than a dozen papers studied this topic, e.g., [12]–[25]. In general, the VNEP and related problems have been in the spotlight for more than 15 years now.

Most existing algorithmic contributions on the VNEP revolve around efficient heuristics which however do not provide any formal approximation guarantees (cf. survey [26] for an overview). Only recently first approximations for variants of it were found [6], [7], [27]. Arguably, one reason for this late discovery is its inherent hardness [1], [28]. Specifically, it was recently shown that the VNEP is NP-complete and inapproximable unless $P = NP$ *under any objective* [1], i.e., including the above-formalized profit and cost cases. Furthermore, it was shown that this hardness is a structural one: even when disregarding capacities altogether and only enforcing the validity of mappings, the VNEP remains hard as long as the structure of request graphs is not too limited. Specifically, it was proven that the hardness even pertains for planar request graphs of bounded degree [1].

Given the complexity results, approximations for general request graph topologies are ruled out (unless $P = NP$) and approximations for *restricted* classes of request graphs were presented only recently [6], [7], [27]. Specifically, while Even et al. proved approximations for embeddings of chains in [27] and Rost and Schmid proved approximations for cactus request graphs [6], in [7] the first approximations for *arbitrary* request topologies were derived. This most recent approximation result, however, comes at the price of being polynomial-time only for graphs of *bounded treewidth*, i.e., graphs which exhibit a certain closeness to trees. All of the known approximations above share the following properties: (i) they are based on the randomized rounding of linear programming solutions, (ii) violate capacities by certain factors in the general setting, and (iii) apply only to the profit variant of the VNEP.

To the best of our knowledge, the only cost approximation result (cf. Definition 5) on VNEP is by Bansal et al. [29]. While the main objective and result of that paper is a very different one, their solution includes a cost approximation result. However, the results are limited to trees only.

VII. Conclusion

We revisited the general Virtual Network Embedding Problem and presented a first constant approximation algorithm for the embedding cost, for a model with and for a model without routing flexibilities. Our algorithm is not only interesting in theory (the first approximation algorithm of its kind), but also performs well in simulations.

We understand our work as a first step and believe that it opens several interesting avenues for future research. In particular, while it is known that the considered problems are NP-hard, it will be interesting to further explore the optimality of our analytic bounds.

Acknowledgments. Research supported by ERC Consolidator grant AdjustNet (No. 864228).

References

[1] M. Rost and S. Schmid, "Charting the Complexity Landscape of Virtual Network Embeddings," in *IFIP Networking*, 2018.

[2] M. Suter, R. Eidenbenz, Y.-A. Pignolet, and A. Singla, "Fog application allocation for automation systems," in *IEEE Conference on Fog Computing*, 2019.

[3] N. Robertson and P. D. Seymour, "Graph minors. iii. planar tree-width," *Journal of Combinatorial Theory, Series B*, vol. 36, no. 1, pp. 49–64, 1984.

[4] J. Flum and M. Grohe, "Parametrized complexity theory." Springer, 2006.

[5] H. L. Bodlaender, "A partial k-arboretum of graphs with bounded treewidth," *Theoretical computer science*, vol. 209, no. 1-2, pp. 1–45, 1998.

[6] M. Rost and S. Schmid, "Virtual Network Embedding Approximations: Leveraging Randomized Rounding," in *IFIP Networking*, 2018.

[7] M. Rost, E. Döhne, and S. Schmid, "Parametrized complexity of virtual network embeddings: Dynamic & linear programming approximations," in *ACM SIGCOMM Computer Communication Review*. ACM, 2019.

[8] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric algorithms and combinatorial optimization*. Springer-Verlag Berlin Heidelberg, 1988. [Online]. Available: https://doi.org/10.1007/978-3-642-97881-4

[9] Y.-A. Pignolet, S. Schmid, and G. Tredan, "Tomographic Node Placement Strategies and the Impact of the Routing Model," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, p. 42, 2017.

[10] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 9, pp. 1765 –1775, october 2011.

[11] M. Rost and S. Schmid, "Virtual network embedding approximations: Leveraging randomized rounding." Proc. IFIP Networking, 2018.

[12] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Joint service placement and request routing in multi-cell mobile edge computing networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 10–18.

[13] V. Farhadi, F. Mehmeti, T. He, T. La Porta, H. Khamfroush, S. Wang, and K. S. Chan, "Service placement and request scheduling for data-intensive applications in edge clouds," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1279–1287.

[14] S. Pasteris, S. Wang, M. Herbster, and T. He, "Service placement with provable guarantees in heterogeneous edge computing systems," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 514–522.

[15] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1459–1467.

[16] G. Sallam and B. Ji, "Joint placement and allocation of virtual network functions with budget and capacity constraints," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 523–531.

[17] J. Zhang, Z. Wang, C. Peng, L. Zhang, T. Huang, and Y. Liu, "Raba: Resource-aware backup allocation for a chain of virtual network functions," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1918–1926.

[18] Z. Zheng, J. Bi, H. Yu, H. Wang, C. Sun, H. Hu, and J. Wu, "Octans: Optimal placement of service function chains in many-core systems," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 307–315.

[19] P. Kortoçi, L. Zheng, C. Joe-Wong, M. Di Francesco, and M. Chiang, "Fog-based data offloading in urban iot scenarios," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 784–792.

[20] H. Xu, Y. Liu, W. C. Lau, J. Guo, and A. Liu, "Efficient online resource allocation in heterogeneous clusters with machine variability," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 478–486.

[21] N. Shahriar, S. Taeb, S. R. Chowdhury, M. Tornatore, R. Boutaba, J. Mitra, and M. Hemmati, "Achieving a fully-flexible virtual network embedding in elastic optical networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1756–1764.

[22] T. Ouyang, R. Li, X. Chen, Z. Zhou, and X. Tang, "Adaptive user-managed service placement for mobile edge computing: An online learning approach," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1468–1476.

[23] D. Y. Zhang and D. Wang, "An integrated top-down and bottom-up task allocation approach in social sensing based edge computing systems," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 766–774.

[24] J. Meng, H. Tan, C. Xu, W. Cao, L. Liu, and B. Li, "Dedas: Online task dispatching and scheduling with bandwidth constraint in edge computing," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2287–2295.

[25] F. Wang, C. Zhang, J. Liu, Y. Zhu, H. Pang, L. Sun *et al.*, "Intelligent edge-assisted crowdcast with deep reinforcement learning for personalized qoe," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 910–918.

[26] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual Network Embedding: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

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

[28] 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.

[29] N. Bansal, K.-W. Lee, V. Nagarajan, and M. Zafer, "Minimum congestion mapping in a cloud," *SIAM Journal on Computing*, vol. 44, no. 3, pp. 819–843, 2015.