Improved Scalability of Demand-Aware Datacenter Topologies With Minimal Route Lengths and Congestion

Maciej Pacut[a], Wenkai Dai[a,1,*], Alexandre Labbe[b], Klaus-T. Foerster[c], Stefan Schmid[a,d,e]

[a] *Faculty of Computer Science, University of Vienna, Austria*
[b] *ENSTA Paris, Institut Polytechnique de Paris, France*
[c] *TU Dortmund, Germany*
[d] *TU Berlin, Germany*
[e] *Fraunhofer SIT, Germany*

Abstract

The performance of more and more cloud-based applications critically depends on the performance of the interconnecting datacenter network. Emerging reconfigurable datacenter networks have the potential to provide an unprecedented throughput by dynamically reconfiguring their topology in a demand-aware manner. This paper studies the algorithmic problem of how to design low-degree and hence scalable datacenter networks that are optimized toward the current traffic they serve. Our main contribution is a novel network design which provides asymptotically minimal route lengths and congestion. In comparison to prior work, our design reduces the degree requirements by a factor of four for sparse demand matrices. We further show that the problem is already NP-hard for tree-shaped demands, but permits a 2-approximation on the route lengths and a 6-approximation for congestion. We further report on a small empirical study on Facebook traces.

Keywords: Network Design, Congestion, Reconfigurable Topologies, Optical Networks, Algorithms, Complexity

[*] Corresponding author
Email address: `wenkai.dai@univie.ac.at` (Wenkai Dai)
[1] *Postal address:* Waehringer Str. 29, Office 5.46, A-1090 Vienna, Austria

1. Introduction

As the performance of many data-centric and cloud-based applications increasingly depends on the underlying networks, datacenter networks have become a critical infrastructure of our digital society. Indeed, current application trends introduce stringent performance requirements and a demand for datacenter networks providing ultra-low latency and high bandwidth. For example, emerging distributed machine learning applications which use highspeed computational devices, periodically require large data transfers during which the network can become the bottleneck. Another example is today's trend of resource disaggregation in datacenters, which introduces a need for very fast access to remote resources (GPU, memory and disk) [1]. Traces of jobs from a Facebook cluster reveal that network transfers on average account for a third of the execution time [2].

Emerging reconfigurable datacenter topologies, enabled by novel optical technologies, introduce new opportunities to significantly improve datacenter performance. In particular, by dynamically establishing topological shortcuts, reconfigurable datacenter networks allow to overcome the cost (or "tax" [3]) of multihop routing [4, 5], or to improve the flow completion time of elephant flows by directly connecting frequently communicating racks, in a demand-aware manner [6, 7, 8, 9, 10, 11, 12, 5].

Demand-aware networks are particularly motivated by empirical studies showing that communication patterns feature much structure. Indeed, traffic matrices (a.k.a. demand matrices) are often sparse and skewed in datacenters [13, 10, 14]. This introduces optimization opportunities, which stands in stark contrast to traditional, demand-oblivious datacenter network designs [15, 16, 17].

This paper studies a fundamental algorithmic problem underlying such reconfigurable networks: how to design a demand-aware topology which, given a demand matrix, provides short topological routes between frequently communicating nodes (e.g., top-of-rack switches [18]), also minimizing congestion.

In particular, for scalability reasons and as reconfigurable hardware consumes space and power, the interconnecting network should be of *low degree*, ideally a small constant.

1.1. Our Contributions

Our contributions revolve around the design of demand-aware networks (BNDs) under a degree restriction, which asymptotically minimize communication cost and congestion, especially when the demand matrix induces a sparse graph or tree. In particular, we present an algorithm to design a network of maximum degree $3\Delta_{\text{avg}} + 8$ with asymptotically optimal route lengths and congestion, when the demand matrix is induced by a sparse graph of an average degree Δ_{avg}. This reduces the required maximum degree of the network by a factor of $4\times$ compared to the previous work, which is significantly more scalable.

We also show that the demand-aware network design problem is NP-hard, already when ignoring congestion and if both the demand itself and the network topology are restricted to be trees; prior work already established the hardness for general demands [19]. We moreover prove that optimizing for congestion, independent of route lengths, is NP-hard as well. On the positive side, we show that for tree-demands, one can jointly 2-approximate the optimal route lengths and 6-approximate the minimum congestion.

Finally, we provide empirical insights into our approach, considering traffic traces from Facebook.

1.2. Technical Novelty

Our network design algorithm builds upon the ego-tree technique introduced by Avin et al. in [20, 12]. In this approach, the network is designed by first constructing an optimal constant-degree tree for each source node v: v is placed at the root of the tree and its destinations are placed such that more frequent communication partners are closer to the root. As these trees are optimized for a single node, they are called *ego-trees*. The network is then a union of all the ego-trees of individual nodes, postprocessed by an algorithm which reduces the degree while preserving distances [20]. The design is flexible in that it allows for various flavors of ego-trees (e.g., Huffman trees, Mehlhorn trees, etc.).

In this paper we propose a tree called *Round Robin Tree* that is particularly well suited to jointly minimize weighted route length and congestion, and which we can interconnect with other trees in a low-degree manner. Comparing to similar approach [21], our construction significantly improves the approximation ratio of route length from $\log^2(\Delta_{\max} + 1)$ to 2, where $\Delta_{\max}$ is the maximum degree of the designed network.

1.3. Organization

The remainder of this paper is organized as follows. In §2, we introduce the bounded network design (BND) problems. For demands restricted to trees, we study the BND problem for optimizing the communication cost in §3, and then extend the objective to optimize both communication cost and congestion in §4. We design the networks for the sparse demand graphs achieving near-optimal routing lengths in §5, additionally achieving near-optimal congestion. We then discuss the applicability of the proposed solutions under real-world traffic traces in §6. After reviewing related work in §7, we conclude and sketch future research directions in §8.

2. Model

We consider the following general model. Given a set of n nodes $V = \{v_1, \ldots, v_n\}$ (e.g., ToRs, servers, peers, etc.), the *communication requests* over $V \times V$ form a *communication pattern*, which can be modeled by an $n \times n$ matrix $D := (d_{ij})_{n\times n}$. An entry $d_{ij} \in \mathbb{R}_{\geq 0}$ of D indicates the communication quantity from the source $v_i \in V$ to the destination $v_j \in V$. If the matrix D is normalized, each entry d_{ij} of D represents the *communication demand* from v_i to v_j. While demands can be asymmetric, the physical communication links of the network are bi-directed, and we hence represent the communication matrix D as an undirected *demand graph* G_D on the same set of nodes V: a weighted (undirected) graph, where each edge $\{v_i, v_j\} \in E(G_D)$ has a weight $w(v_i, v_j) = d_{ij} + d_{ji}$. In this paper, we are especially interested in fundamental specific graphs for G_D, in particular trees and sparse graphs, i.e., graphs where the number of edges is linear in terms of the number of vertices. Hence, the average degree in sparse graphs is constant. To serve G_D, we need to *design* a network $N(D)$ over the set of nodes V, s.t., for each edge $\{v_i, v_j\} \in E(G_D)$, there must be a (bi-directed) path between v_i and v_j in the *designed network* $N(D)$. When D is clear from the context, we often abbreviate $N(D)$ by N, and we also refer to the designed network as the *host graph*. We consider two fundamental objectives when optimizing the network topology toward the demand. We next discuss them in turn.

2.1. Objective #1: Route Length

The first objective considered in this paper is related to the achieved route lengths, weighted by the amount of traffic. Weighted route lengths correspond to the total communication distance in the network. Given a demand graph G_D and a host graph N, for each demand $\{v_i, v_j\} \in E(G_D)$, let $\mathsf{dist}_N(v_i, v_j)$ be the length of the shortest path between v_i and v_j in N. Then, the cost to serve G_D by a host graph N is defined as:

$$\mathsf{cost}(G_D, N) = \sum_{\{v_i, v_j\} \in E(G_D)} w(v_i, v_j) \cdot \mathsf{dist}_N(v_i, v_j).$$

Given a demand graph G_D over V, let $\mathcal{N}$ denote a collection of host graphs N over V, where each $N \in \mathcal{N}$ can serve G_D. Usually, we are only interested in some host graphs of $\mathcal{N}$ which satisfy specific desired properties. In particular, we are interested in scalable topologies of low degree, and henceforth, let $\mathcal{N}_\Delta \subseteq \mathcal{N}$, where $\Delta \in \mathbb{N}^+$, denote all host graphs $N \in \mathcal{N}$ that have their maximum degree bounded by Δ. Meanwhile, our design objective is to find a host graph $N \in \mathcal{N}_\Delta$ of bounded degree to minimize the communication cost of serving G_D.

Definition 1 (Bounded Network Design (BND)). *Given a communication matrix D, i.e., a demand graph G_D, and a degree bound Δ, find a host graph*

$N \in \mathcal{N}_\Delta$ that minimizes the communication cost of serving D:

$$\mathit{BND}\left(G_D, \Delta\right) = \min_{N \in \mathcal{N}_\Delta} \textsf{\textit{cost}}\left(G_D, N\right).$$

2.2. Objective #2: Congestion

We are further interested in the congestion experienced in the network, which can negatively affect network performance. When only considering distances, the routing itself is easy to optimize on a given host graph, by considering the shortest paths. In the context of congestion, it can be advantageous to define different routing paths via a *routing scheme* $\Gamma(N)$ for a network N.

In the following, we consider the practically important model of unsplittable flows, i.e., each demand is routed along a single path. In more detail, a routing scheme $\Gamma(N)$ for N is defined by a set of simple paths $\Gamma_{v_i v_j}$, one for each pair of nodes v_i, v_j. Herein, a demand between v_i, v_j imposes a load of $w(v_i, v_j)$ on each edge $e \in \Gamma_{v_i v_j}$ it is routed on, where the congestion $\textsf{cong}$ is defined by the most loaded edge:

$$\textsf{cong}\left(G_D, \Gamma(N)\right) = \max_{e \in \Gamma(N)} \sum_{\substack{e \in \Gamma_{v_i v_j} \\ \Gamma_{v_i v_j} \in \Gamma(N)}} w(v_i, v_j).$$

Ideally, we want to find a host graph and routing scheme that minimizes the congestion in the network:

Definition 2 (Congested Bounded Network Design ($\textsf{cong}$-BND))**.** *Given a demand graph G_D, and a degree bound Δ, find a host graph $N \in \mathcal{N}_\Delta$ and routing scheme $\Gamma(N)$ that minimizes the congestion of serving D:*

$$\textsf{\textit{cong}}\text{-}\mathit{BND}\left(G_D, \Delta\right) = \min_{N \in \mathcal{N}_\Delta, \Gamma(N)} \textsf{\textit{cong}}\left(G_D, \Gamma\left(N\right)\right).$$

2.3. Joint Optimization

Next, we define a problem of optimizing both communication distances (in terms of route length) and congestion at the same time. To this end, we first analogously specify the communication cost incurred by routing along $\Gamma(N)$, i.e., $\mathsf{cost}\left(G_D, \Gamma(N)\right)$:

$$\mathsf{cost}\left(G_D, \Gamma(N)\right) = \sum_{\{v_i, v_j\} \in E(G_D)} w\left(v_i, v_j\right) \cdot \mathsf{dist}_{\Gamma(N)}\left(v_i, v_j\right).$$

Given a demand graph G_D and a network N, we use $\mathsf{cost}\text{-BND}^*\left(G_D, \Delta\right)$ and $\mathsf{cong}\text{-BND}^*\left(G_D, \Gamma(N)\right)$ to denote the optimal route lengths and congestion respectively. While some networks and demands permit solutions that are optimal both in route lengths and congestion, most commonly this is not the case. We hence define the joint design problem as proposed by Avin et al. [21]:

Definition 3 ((c,d)-Bounded Network Design ((c,d)-BND))**.** *Given a demand graph G_D, and a degree bound Δ, the (c,d)-BND problem is to find a host graph $N \in \mathcal{N}_\Delta$ and routing scheme $\Gamma(N)$ s.t. both communication cost and congestion are bounded w.r.t. individual objective function optimization, i.e.:*

$$\mathsf{cong}\left(G_D, \Gamma(N)\right) \leq c \cdot \mathsf{cong}\text{-}BND^*\left(G_D, \Delta\right) + c',$$

$$\mathsf{cost}\left(G_D, \Gamma(N)\right) \leq d \cdot \mathsf{cost}\text{-}BND^*\left(G_D, \Delta\right) + d',$$

with constants c', d' being problem-independent parameters.

3. Route Lengths in Trees

In this section, we study the scenario where the demand graph is a tree, and the goal is to minimize the route lengths. We start with notations and preliminaries in §3.1. Then, in §3.2 we study optimal topologies for the special case of 2-level trees (i.e., stars) and we provide lower bounds and optimal constructions. We then show in §3.3 how to efficiently compose solutions for the general tree case from the optimal solutions for 2-level trees using the ego-tree design (cf. §1.2). Lastly, we illustrate the intractability of the BND problem when restricting the demand graphs to be a tree in §3.4.

For simplicity of presentation, we consider undirected tree demands, and we note that the results can be extended to directed tree demands. We defer the discussion about congestion to the next section (§4).

3.1. Preliminaries

For a tree T rooted at r, we define a *level* of T as all nodes with equal distance to r. Let the root r constitute the first level of T, then the set of nodes having a distance i to the root r, where $i \in \mathbb{N}$ and $i \geq 1$, defines *the* $(i+1)$*-th level* of T. A tree containing k different levels, where $k \in \mathbb{N}$ and $k \geq 1$, is called a k-level tree, e.g., a tree consisting of a root and its children is a 2-level tree. We note that each tree has a unique partition into levels. In principle, then the root r can be thought as the most common node of intensive traffic among the nodes of T.

For integers $\alpha, \beta \geq 2$ we define a (α, β)*-ary tree* as a tree where the maximum degree of the root is α, and the maximum degree of every non-root node is β.

Let T_D denote the demand tree on the set of n nodes V, rooted at $r \in V$. Given a 2-level demand tree T_D on nodes V, let $\vec{V}$ be a sequence of nodes from V sorted in non-increasing order according to their edge weights to the root $\vec{V}[1] = r$ (the first element). Let $T_D\left(v\right)$ be a *subtree* of T_D induced by an inner-node $v \in V$ and its children in T_D, then $T_D\left(v\right)$ is a 2-level tree. Our algorithm produces a tree network, thus we refer to the network as a *host tree*, and to the part constructed for $T_D(v)$ a *local host tree* for node v.

3.2. Locally Optimal Trees

We present a gadget[2](α, β)-$\mathsf{LocalTree}\left(T_D\right)$ for constructing a local host tree for a given 2-level tree T_D and integers $\alpha, \beta \geq 2$. We define $\mathsf{LocalOpt}(T_D, \Delta) =$

[2]Here and in the following, the term *gadget* refers to an auxiliary graph construction.

$(\Delta, \Delta-1)$-$\mathsf{LocalTree}\,(T_D)$, and in Lemma 1 we show it is optimal in terms of route lengths.

The (α,β)-$\mathsf{LocalTree}\,(T_D)$ is constructed as follows. Let T_D be a 2-level demand tree on n nodes V, rooted at r. The (α,β)-$\mathsf{LocalTree}\,(T_D)$ constructs an (α,β)-ary tree containing the nodes V. Let the sequence $\vec{V}$ of nodes V consist of $\vec{V}[1] = r$ and followed by sorted children of r in a non-increasing order of their edge weights in T_D. To construct the (α,β)-$\mathsf{LocalTree}\,(T_D)$, we start with an empty tree, and we insert the nodes of $\vec{V}$ sequentially. To insert a node $\vec{V}[i]$, we attach it at an arbitrary place closest to the root, without violating the degree constraints of (α,β)-ary tree. Note that the constructed network is a balanced tree.

Lemma 1. *If a demand tree T_D on V is a 2-level tree, then $\mathsf{LocalOpt}\,(T_D,\Delta)$ is an optimal solution to BND$\,(T_D,\Delta)$.*

Proof. Let $T = \mathsf{LocalOpt}\,(T_D,\Delta)$ and let T^* denote an optimal host tree of BND$\,(T_D,\Delta)$, rooted at r.

For each level of T, the tree T^* cannot contain more nodes at the same level than T, except for the last level (otherwise, the degree bound Δ cannot be satisfied in T^*). If T^* would contain less nodes on a (non-last) level, then the solution can be improved, a contradiction. Thus, the corresponding levels of T and T^* contain the same number of nodes.

Assume that the cost of T^* is strictly smaller than the cost of T. Let m_i (resp. m^*_i) denote the sum of demands to r of nodes at the level i of T (resp. T^*). Let j be the first level where $m_j \neq m^*_j$. Since T assigns the largest weights closest to the root, the value m_j is maximal. Thus, $m^*_j < m_j$, and since j is the first such level, there exists a node $u_{j'}$ in a level $j' > j$ with higher demand to r than a node u_j. Swapping them reduces the cost of T^*, thus it is not optimal, a contradiction.

Finally, T is a feasible solution. It is a $(\Delta,\Delta-1)$-ary tree and the degree bound is Δ. □

Note that it is impossible to construct a feasible solution from these locally optimal topologies, as we cannot combine them without violating the degree constraints. However, although the union of $\mathsf{LocalOpt}$ gadgets is not a feasible topology, sum of costs of routing demands in them constitutes a valid lower bound on the cost of optimal solution. We use this lower bound in the proof of Lemma 2.

3.3. Approximation Algorithm

We now investigate the general scenario where the height of the demand tree is arbitrary. To this end, we first provide the details of an approximation algorithm, and then in Theorem 1 we show that the algorithm is a 2-approximation for every $\Delta \geq 5$. The algorithm is an efficient version of ego-tree design (cf. §1.2), accounting for the tree structure of the input.

The algorithm for general demand trees is defined as follows. We are given a demand tree T_D on n nodes V, with the root $r \in V$ and a degree bound Δ. For each inner-node $v^* \in V$, the subtree $T_D(v^*)$ of T_D (a subtree induced by v^* and its children), is a 2-level tree rooted at v^*. For each subtree $T_D(v^*)$, $\mathsf{ALG}(T_D, \Delta)$ constructs a local host tree T_{v^*} rooted at v^* for $\mathrm{BND}(T_D(v^*), \Delta)$ using (α^*, β^*)-$\mathsf{LocalTree}(T_D(v^*))$. Throughout this section, we use $\alpha^* = \lfloor (\Delta - 1)/2 \rfloor$ and $\beta^* = \lceil (\Delta - 1)/2 \rceil$. The tree T_{v^*} is (α^*, β^*)-ary.

For each child u of the node v^*, $\mathsf{ALG}(T_D, \Delta)$ preserves the degree of α^* for u since there could be another local host tree T_u rooted at u to serve requests defined by $T_D(u)$. And the local host tree T_u must be connected with T_{v^*} by joining two identical nodes u. After joining two local host trees, the node u would have $\alpha^* + \beta^* \leq \Delta - 1$ children, and the final degree of u does not exceed Δ. The algorithm $\mathsf{ALG}(T_D, \Delta)$ terminates after all inner-nodes are processed and returns the tree $\mathsf{T}_{\mathrm{out}}$ as the host tree for the problem $\mathrm{BND}(T_D, \Delta)$.

We next show that the above procedure achieves 2-approximation: using the local host tree (α^*, β^*)-$\mathsf{LocalTree}(T_D(v^*))$, the distance from an inner-node $v^* \in V$ to a child of v^* is at most twice its distance in the local optimal host tree $\mathsf{LocalOpt}(T_D(v^*), \Delta)$.

Lemma 2. *For a 2-level tree T_D, the network (α^*, β^*)-$\mathsf{LocalTree}$ $(T_D(v^*))$ is a 2-approximation for BND (T_D, Δ) for every $\Delta \geq 5$.*

Proof. Given a 2-level demand tree T_D rooted at r, and a degree bound Δ, let T_1 be an optimal host tree of $\mathrm{BND}(T_D, \Delta)$, then we claim that the host tree $T_2 = \mathrm{ALG}(T_D, \Delta)$ has $\mathsf{cost}(T_D, T_2) \leq 2 \cdot \mathsf{cost}(T_D, T_1)$ for $\Delta \geq 5$.

Consider an optimal host T_1 computed by $\mathsf{LocalOpt}(T_D, \Delta)$ (see Lemma 1). Then, the claim of the lemma is equivalent to the claim: for any node $v \in V \setminus \{r\}$, if v has $\mathsf{dist}_{T_1}(v, r) = k$, where $k \geq 1$, then $\mathsf{dist}_{T_2}(v, r) \leq 2k$.

Let $v \in V \setminus \{r\}$ be an arbitrary node having the distance k to the root r on T_1, where $k \geq 1$, then v is on the $(k+1)$-th level of T_1. Let j' denote the total number of nodes contained on the first $k+1$ levels of T_1, where $j' \leq n$. $\mathsf{LocalTree}$ assigns nodes to levels in an order of non-increasing demands, thus $v \in \{\vec{V}[1], \ldots, \vec{V}[j']\}$. Let j denote the total number of nodes placed on the first $2k+1$ levels of T_2, where $j \leq n$. If v is also contained in $\{\vec{V}[1], \ldots, \vec{V}[j]\}$, then it implies $\mathsf{dist}_{T_2}(v, r) \leq 2k$ directly. It remains to show $j' \leq j$. We assume $j' < n$ and $j < n$, otherwise $j' \leq j$ is established by $j' = n = j$. First, by the definition of j', we bound j' by the following inequality:

$$j' \leq 1 + \Delta \cdot \frac{(\Delta - 1)^k - 1}{\Delta - 2}.$$

When $\Delta = 5$, it holds that $j' \leq 1 + \frac{5}{3} \cdot \left(4^k - 1\right)$.

If $\Delta \geq 6$, an upper bound of j' is derived as follows:

$$j' \leq 1 + \Delta \cdot \frac{(\Delta - 1)^k - 1}{\Delta - 2} \leq 1 + 1.5 \cdot \left((\Delta - 1)^k - 1\right) \leq 1.5 \cdot (\Delta - 1)^k .$$

The formula for j is the following:

$$j = 1 + \left\lfloor \frac{\Delta - 1}{2} \right\rfloor \cdot \frac{\left\lceil \frac{\Delta - 1}{2} \right\rceil^{2k} - 1}{\left(\left\lceil \frac{\Delta - 1}{2} \right\rceil - 1 \right)} .$$

When $\Delta = 5$, we have $j = 1 + 2 \cdot \left(2^{2k} - 1\right)$. Moreover, for $\Delta \geq 6$, we note that the following inequality holds.

$$\left\lfloor \frac{\Delta - 1}{2} \right\rfloor \geq \left\lceil \frac{\Delta - 1}{2} \right\rceil - 1, \quad \text{if } \Delta \geq 6 .$$

We derive a lower bound for j under conditions $\Delta \geq 6$ and $1.5^k \cdot (\Delta - 1)^k$.

$$j = 1 + \left\lfloor \frac{\Delta - 1}{2} \right\rfloor \cdot \frac{\left\lceil \frac{\Delta - 1}{2} \right\rceil^{2k} - 1}{\left(\left\lceil \frac{\Delta - 1}{2} \right\rceil - 1 \right)} \geq 1 + \left(\left\lceil \frac{\Delta - 1}{2} \right\rceil^{2k} - 1 \right)$$

$$\geq 1 + \left(\left(\frac{\Delta - 1}{2} \right)^k \cdot \left\lceil \frac{\Delta - 1}{2} \right\rceil^k - 1 \right) \geq \left(\frac{\Delta - 1}{2} \right)^k \cdot 2^k \cdot \left(\frac{3}{2} \right)^k$$

By above inequalities, we have $j \geq j'$ for $\Delta \geq 5$. This implies $\mathsf{dist}_{T_2}(v, r) \leq 2 \cdot \mathsf{dist}_{T_1}(v, r)$. □

Finally, we combine bounds for local host trees to show a 2-approximation for arbitrary tree demands.

Theorem 1. *For a demand tree T_D, the $\mathsf{ALG}(T_D, \Delta)$ is a 2-BND(T_D, Δ) for every $\Delta \geq 5$.*

Proof. First, we show that the solution $\mathsf{T}_{\text{out}} = \mathsf{ALG}(T_D, \Delta)$ is feasible, i.e., it respects the degree bound Δ. Each node $v \in V$ participates in its own LocalTree and the LocalTree of its parent in T_D. Its degree in its own LocalTree is bounded by α^*, since it v is the root. Its degree in its parent's LocalTree is bounded by β^*, the number of its children (v is an inner node) plus at most one edge towards its parent. Thus, the total degree is $\alpha^* + \beta^* + 1 \leq \Delta$.

Now, we prove the approximation ratio. For each subtree $T_D(v^*)$ of T_D, the algorithm $\mathsf{ALG}(T_D, \Delta)$ computes a local host tree T_{v^*}. Joining a local host tree T_{v^*} into T_{out} cannot increase the communication cost for T_{v^*}. Thus, for the set of all inner nodes V^*, we have

$$\sum_{v^* \in V^*} \mathsf{cost}\left(T_D(v^*), T'_{v^*}\right) \leq \mathsf{cost}\left(T_D, T_{\text{opt}}\right),$$

and by Lemma 2, the solution is a 2-approximation for $\text{BND}(T_D, \Delta)$ for $\Delta \geq 5$. □

We note that the running time of our algorithm ALG is dominated by sorting and can hence be bounded by $O(n \log n)$.

3.4. Computational Complexity

We next investigate the computational complexity of minimizing the communication cost when the demand-aware network must be a tree. Prior work showed this problem to be NP-hard for general demand graphs [19]. We go beyond and show the NP-hardness of the BND problem even if *both* the given demand graphs and the returned host graph required to be trees. To this end we perform a reduction from the 3-Partition problem [22], namely:

Definition 4 (3-Partition[22])**.** *Given a finite set A of $3m$ elements, a bound $B \in \mathbb{Z}^+$, and a size function: $s(a) \in \mathbb{Z}^+$ for each $a \in A$ such that each $s(a)$ satisfies $K/4 < s(a) < K/2$ and such that $\sum_{a\in A} s(a) = mK$, can we partition A into m disjoint sets $A_1, \ldots, A_m$, such that for $1 \le i \le m$, $\sum_{a\in A_i} s(a) = K$, where $|A_i| = 3$?*

Theorem 2. *The BND problem is strongly NP-hard even if both demand graph G_D and host graph N_Δ are restricted to be trees.*

Proof. We prove the claim by reducing from the 3-*Partition* problem (Definition 4 [22]). Given an instance $I = (A, s, m, K)$ of 3-Partition problem, we construct an instance $I' = (G_D, \Delta)$ of the BND problem, where the demand graph G_D and the host graph N_Δ are required to be trees respectively and Δ denote the degree-bound on N_Δ. For the demand graph G_D, we construct a tree $T_D = (V, E_D)$ rooted at a node $r \in V$ s.t., if an edge $\{u, v\} \in E_D$ then we have a demand $D(u, v) > 0$. Let the given degree-bound Δ be m. We define two large constants: α and γ, s.t.,

$$\begin{aligned} \alpha &> 3m \cdot \gamma + m \cdot \big((m-1)^2 - K\big) \\ \gamma &> m \cdot \big((m-1)^2 - K\big) > 1 \,. \end{aligned}$$

The root r has m children: $R = \{r_i : i \in \{1, \ldots, m\}\}$ in T_D, where there is a demand $D(r, r_i) = \alpha$ for each child $r_i \in R$. For each element $a_i \in A$, the root r also has a child $b_i \in B$ with a demand $D(r, b_i) = \gamma$ in T_D, meanwhile each node b_i has $s(a_i)$ children $B_i = \big\{b_{i,1}, \ldots, b_{i,s(a_i)}\big\}$, s.t., each $b_{i,j} \in B_i$ has a demand

$D\left(b_i, b_{i,j}\right) = \alpha$ for each $j \in \{1, \ldots, s(a_i)\}$. Moreover, each node $r_i \in R$ has $m-4$ children $R_i = \{r_{i,1}, \ldots, r_{i,m-4}\}$ in T_D, where each child $r_{i,j} \in R_i$ has a demand $D\left(r_i, r_{i,j}\right) = \alpha$, and also $(m-1)^2 - K$ children $R'_i = \left\{r'_{i,1}, \ldots, r'_{i,(m-1)^2-K}\right\}$. We complete the construction of T_D by giving a demand $D\left(r_i, r'_{i,j}\right) = 1$ for each node $r'_{i,j} \in R'_i$ in T_D.

Let β denote the communication cost of the given demand tree T_D. Moreover, we define another constant β_2 as follows:

$$\beta_2 = 3m \cdot \gamma + m \cdot \left((m-1)^2 - K\right) \ .$$

We claim that the instance I has a valid 3-Partition solution $A^*_1, \ldots, A^*_m$ iff the BND instance I' can be mapped into a host tree $T^* = (V, E^*)$, s.t., its communication cost satisfies

$$\mathsf{cost}\left(T_D, T^*\right) - \beta \leq \beta_2 \ .$$

By observing this construction, it is easy to note that if an edge $\{u, v\} \in E_D$ has $D(u, v) = \alpha$ in T_D, then it must have $\{u, v\} \in E^*$, otherwise $\mathsf{cost}\left(T_D, T^*\right) - \beta \geq \alpha > \beta_2$ since $\mathsf{dist}_{T^*}\left(u, v\right) \geq 2$. Since $\Delta = m$ and $\forall r_i \in R : D(r, r_i) = \alpha$, then these m nodes in R must stay on the second layer of the host tree T^*, where the root r is the first layer of T^*. Moreover, for each node $r_i \in R$, since each child $r_{i,j} \in R_i$ of r_i has $D(r_i, r_{i,j}) = \alpha$, then $\{r_i, r_{i,j}\} \in E^*$. Since R are already placed on the second layer of T^*, then each node $r_{i,j} \in R_i$ has to stay on the third layer of T^* as a child of $r_i \in R$, which has $\mathsf{dist}_{T^*}\left(r_{i,j}, r\right) = 2$. Since each node $b_i \in B$ has $D\left(r, b_i\right) = \gamma$, then we know $\mathsf{dist}_{T^*}\left(r, b_i\right) \leq 2$ for each $b_i \in B$, where $|B| = 3m$, otherwise it must imply

$$\mathsf{cost}\left(T_D, T^*\right) - \beta \geq 3m \cdot \gamma + \gamma > \beta_2 \ .$$

Thus, all nodes of B should be on the third layer of T^*. Due to $\Delta = m$, there are at most $m\left(m-1\right)$ nodes on the third layer of T^*. Since $\bigcup_{r_i \in R} R_i$ contains $m\left(m-4\right)$ nodes, then $3m$ nodes in B can be placed on the third layer T^*. Now, we consider how to partition B into m sets s.t. every three distinct nodes from B become the children of a node $r_i \in R$.

If I has a valid 3-Partition solution $A_1^*, \ldots, A_m^*$, then for each $\{a_j, a_k, a_l\} = A_i^*$, we will place the corresponding nodes $\{b_j, b_k, b_l\}$ as the children of the node r_i. By the definition of the 3-Partition problem, we have $s(a_j) + s(a_k) + s(a_l) = K$; Thus, after placing nodes of B_i exactly under their parent b_i on the fourth layer of T^*, where $i \in \{j, k, l\}$, we can still place $(m-1)^2 - K$ nodes of R_i' under the subtree of r_i on the fourth layer of T^*. Therefore, for each node $r_{i,j}'$ in R_i', we know $\mathsf{dist}_{T^*}(r_{i,j}', r_i) \leq 2 \cdot \mathsf{dist}_{T_D}(r_{i,j}', r_i)$, which implies the communication cost on T^* is increased by at most $(m-1)^2 - K$. It further implies $\mathsf{cost}(T_D, T^*) - \beta \leq \beta_2$.

Conversely, if the instance I does not have a valid 3-Partition solution, then there must exist a set

$$A_i^* = \{a_j, a_k, a_l\}, \text{ s.t.}, \sum_{i \in \{j,k,l\}} s(a_i) > K \ .$$

In the instance I', for any node $r_i \in R$, if we put the nodes b_j, b_k, and b_l, which correspond to the elements $\{a_j, a_k, a_l\}$ of I, as children of r_i in T^*, then there must be at least one node $r_{i,f}' \in R_i'$ that has to be placed two hops far from r_i in T^*, which directly implies $\mathsf{cost}(T_D, T^*) - \beta \geq \beta_2 + 1$. □

4. Routing and Congestion in Trees

We next modify the algorithm for tree demands (§3.3) to jointly minimize the route lengths and congestion. We show a 6-approximation algorithm for congestion (§4.2), while maintaining a 2-approximation for route lengths.

We improve over the approach in [21], that designed an ego-tree with the objective of minimizing the congestion. Their work shows an interesting connection between congestion and scheduling for identical machines, and we refine this approach by improving guarantees for route lengths.

We propose a construction called *Round Robin Trees*, a novel gadget for connecting a node to its neighbors. It allows to design a network for tree demands that is a 2-approximation for route lengths and a 6-approximation for congestion (a $(6,2)$-BND). To this end, first we show an algorithm Sorted Round Robin, a

2-approximation for scheduling for identical machines (§4.1) with desired properties for this application. Next, we use this scheduling algorithm internally in the construction of Round Robin Trees (§4.2).

4.1. A Connection to Scheduling

The scheduling on identical machines problem is defined as follows. We are given a set of n jobs J and m identical machines $M_1, M_2, \ldots, M_m$. The goal is to minimize the makespan of the schedule (the total processing time of the most loaded machine).

For scheduling on identical machines, a simple 4/3-approximation algorithm was proposed by Graham in his classic work on scheduling [23]. The algorithm called Longest Processing Time First (LPTF) examines the jobs in non-increasing duration order, and assigns each job to the currently least loaded machine.

LPTF aims at balancing the load between machines, without considering the number of jobs assigned to machines. LPTF may assign a large *number* of jobs to a single machine. This property is undesirable, as using such an algorithm in network design results in increased route lengths [21].

To preserve the 2-approximation for route lengths (proved in Section §3), we devise a different constant approximation algorithm for scheduling on identical machines, which balances both the load and the number of jobs assigned to machines.

Algorithm Sorted Round Robin. Sort the set of jobs in non-increasing duration order. Assign job i to machine $M_{(i \mod n)+1}$.

Theorem 3. *Sorted Round Robin is a* 2*-approximation for scheduling on identical machines.*

Proof. Let ALG denote the Sorted Round Robin algorithm. Let $j_1, j_2, \ldots, j_n$ be the set of jobs sorted in non-increasing duration order. For $i \in \{1, 2, \ldots, m\}$, let ℓ_i denote the total duration of jobs assigned by ALG to the machine M_i.

First, we upper bound the cost of ALG. ALG assigns the jobs evenly among machines in order of non-increasing duration, starting from the machine M_1.

Thus, $\ell_1 \geq \ell_2 \geq \ldots \geq \ell_m$, and additionally the makespan of the algorithm is ALG $= \ell_1$.

In ALG's machine M_1, there exists a job of duration j_1 (ALG starts by assigning the longest job to the machine M_1). Thus, ALG assigns to M_1 a set of jobs (possibly empty) $b_1, b_2, \ldots, b_p$ of total load B so that $B = \ell_1 - j_1$.

The crucial observation is that assigning each item b_i to M_1 entails assigning one item no smaller than b_i to each of $M_2, M_3, \ldots, M_m$. This holds as $n-1$ jobs assigned just before b_i were assigned to the machines $M_2, M_3, \ldots, M_m$, and each of these jobs was no shorter than b_i. Let C be the total duration of jobs assigned by ALG to machines $M_2, M_3, \ldots, M_m$. Than, $C \geq \sum_{i=1}^{p} b_i(m-1) = B(m-1)$.

To bound the approximation ratio, we consider two cases.

1. Consider the case $B < j_1$. Then ALG $= j_1 + B < 2j_1$. The makespan of OPT is lower-bounded by j_1, the duration of the longest job. Thus, in this case ALG/OPT < 2.

2. Consider the case $B \geq j_1$. OPT is lower-bounded by $\frac{1}{m}\sum_{i=1}^{m} j_i$ (this can be seen as a lower bound that allows splitting the jobs between machines). This lower bound can be expressed as OPT $\geq \frac{1}{m}(j_1 + B + C) > \frac{1}{m}(B + C)$. Thus,
$$\frac{\text{ALG}}{\text{OPT}} \leq \frac{j_1 + B}{\frac{1}{m}(B+C)} \leq \frac{mj_1 + mB}{B + (m-1)B} \leq 2,$$
where the last inequality follows by case assumption $B \geq j_1$.

□

4.2. Our Gadget: Round Robin Trees

For any integer α, β, an (α, β)-Round Robin Tree for a given 2-level tree rooted at r is constructed as follows. First, construct an (α, β)-ary tree. To assign nodes, we construct a scheduling instance with α identical machines. For each edge (r, u) with demand d, we construct a job j_u with duration d. Then, we run the Sorted Round Robin scheduling algorithm from Section §4.1. We observe the job assignment, and for each machine M_i with jobs $J_i = \{j_{u_1}, j_{u_2}, \ldots, j_{u_w}\}$, we assign the nodes $\{u_1, u_2, \ldots, u_w\}$ to the i-th subtree of the Round Robin Tree. Precisely, to decide on the placement of nodes within a subtree, we consider the nodes in non-increasing demand weight order; we place each node at an arbitrary unoccupied place closest to the root of the subtree.

We emphasize similarities of Round Robin Trees to LocalTree (§3.2). For $\alpha^* = \lfloor (\Delta - 1)/2 \rfloor$ and $\beta^* = \lceil (\Delta - 1)/2 \rceil$, an (α^*, β^*)-Round Robin Trees is a variant of LocalTree, where we concretize the arbitrary order of assigning nodes to levels during the construction of LocalTree. To decide on the ordering, we run the scheduling algorithm from §4.1.

An (α^*, β^*)-Round Robin Trees achieve a 2-approximation for route length, which is a significant improvement in comparison to prevously proposed ego-tree variants [21] with the route length approximation of $\log^2(\Delta_{\max} + 1)$, where $\Delta_{\max}$ is the maximum degree of the designed network. The tradeoff is that the congestion approximation ratio for 2-level trees grows: the previous ego-tree variants were proved to be 4/3-approximation, and a Round Robin Tree is a 2-approximation. This is due to using different scheduling algorithms: previous ego-tree variants uses LPTF algorithm, and Round Robin Trees uses Sorted Round Robin (§4.1).

4.3. A 6-Approximation Algorithm

We now present a variant of the algorithm from §3, and prove that it guarantees a 6-approximation for congestion, while maintaining 2-approximation for route lengths for tree-induced demands.

Lemma 3. *Let $\mathcal{D}$ be a tree. Then, there exists a $(6, 2)$-BND($\mathcal{D}$).*

Proof. Let $\alpha^* = \lfloor (\Delta - 1)/2 \rfloor$ and $\beta^* = \lceil (\Delta - 1)/2 \rceil$. To achieve the bound on congestion, we use the algorithm from Section §3, where we use (α^*, β^*)-Round Robin Trees instead of LocalTree.

The (α^*, β^*)-Round Robin Tree is equivalent to LocalTree from Section §3 in terms of route lengths: the analysis in Lemma 2 holds for (α^*, β^*)-Round Robin Trees.

Following the arguments of [21], for 2-level demand tree the highest congestion is on the links adjacent to the root. Precisely, the congestion on the link to i-th subtree is equal to the load of the machine M_i. The optimal congestion for general trees is lower-bounded by the solution to a scheduling problem with Δ identical machines [21]. Our solution uses $\alpha*$ machines, thus its approximation ratio is $2 \cdot \Delta/\alpha^* \leq 6$ in comparison to the scheduling instance with Δ machines.

The produced network is a tree, thus routes are unique. Moreover, the links within each LocalTree are used only for routing demands between one node and its children, thus the congestion of the network is the maximum congestion among LocalTree gadgets. Thus, we obtain a 6-approximation for congestion. □

5. Network design for sparse demands

In this section, we present an algorithm that designs a network of maximum degree $3\Delta_{\text{avg}} + 8$ with near-optimal route lengths for sparse traffic patterns of average degree Δ_{avg}.[3] This improves upon previous results of Avin et al. [20] that required a maximum degree of $12\Delta_{\text{avg}}$ to achieve the same bound on route lengths.

The idea behind the algorithm of Avin et al. is the following. First, the algorithm identifies high degree nodes (defined as at least $2\Delta_{\text{avg}}$ degree). To reduce their degree, the algorithm replaces edges between high degree nodes with 2-hop routes through an intermediate vertex. The intermediate vertex is called a *helper node*, and for each replaced edge the algorithm chooses the helper node arbitrarily among low degree nodes (defined as non-high degree nodes). The algorithm assigns each low degree node as a helper for at most $2\Delta_{\text{avg}}$ edges.

Second, the algorithm replaces the edges incident with each high degree node with just one edge that connects the high degree node to a Mehlhorn tree [24] that contains all its neighbors. The Mehlhorn tree is a near-optimal binary

[3]Recall that the sparseness implies that Δ_{avg} is a constant.

search tree with respect to any tree topology (not necessarily a search tree), and using it guarantees near-optimal route lengths [24].

The Mehlhorn tree is a binary tree, thus each node contained in a Mehlhorn tree increases its degree by at most 3. A helping node participates in Mehlhorn trees of high degree node's neighbors instead of the high degree node itself.

5.1. Improved Algorithm for Sparse Demands

We design an algorithm with improved bound for the maximum degree from $12\Delta_{\text{avg}}$ to $3\Delta_{\text{avg}} + 8$. The route lengths bound remain are equal to the bounds of Avin et al. The result provided in [12] shows that using Mehlhorn trees gives asymptotically optimal route lengths when Δ_{avg} is constant.

We modify the algorithm of Avin et al. in the following way. Instead of splitting nodes into high and low degree nodes, our algorithm may choose an *arbitrary* node as a helper for an edge, including the incident node itself. (Helping its own edge is a special case in the algorithm.)

Theorem 4. *Let $\mathcal{D}$ be a sparse demand traffic graph with average degree of Δ_{avg}. Then there exists a $O(1)$-BND($\mathcal{D}$, $\Delta_{\max}$) with maximum degree $\Delta_{\max} = 3\Delta_{\text{avg}} + 8$.*

Proof. We design the network in two stages. First construct an *auxiliary graph* G' that is initially equal to the demand distribution graph $\mathcal{D}$, and we modify certain edges in G'. Then, we construct the network N based on G' by organizing neighbors in a tree (similarly to the LocalTree from Section §3). Finally, we argue that N has the claimed properties.

The algorithm arbitrarily assigns a node to *help* each edge in the demand graph. While doing so, it ensures that each node helps at most $\beta = \lceil \Delta_{\text{avg}}/2 \rceil$ edges. We construct the auxiliary graph G' based upon the helper nodes assignment. Consider an edge (i, j) and an arbitrary helping node k. If k is chosen as either i or j, then we do not modify any edges of G'. Otherwise, if k is neither

i nor j, we replace the edge (i, j) in G' with 2-hop path through k:

$$p(i, j) = 0$$
$$p(i, k) = p(i, k) + p(i, j)$$
$$p(k, j) = p(k, j) + p(i, j)$$

We say that the edges (i, k) and (k, j) added to (and from) intermediate nodes are *intermediate edges*.

Next, we construct the network N based upon the auxiliary graph G'. We start with an empty network N. In G', a node i has two types of new neighbors: the set G_i of intermediate nodes that replaced initial edges of i, and the set H_i of nodes in whose edges i is helping. Among G_i we distinguish the set G_i^- (resp. G_i^+) of nodes that are connected with i with an incoming (resp. outgoing) edges. For each node i, the algorithm constructs two Mehlhorn trees in N, one for G_i^- and another for G_i^+, and connects its roots to i. An example of the construction is depicted in Figure 1.

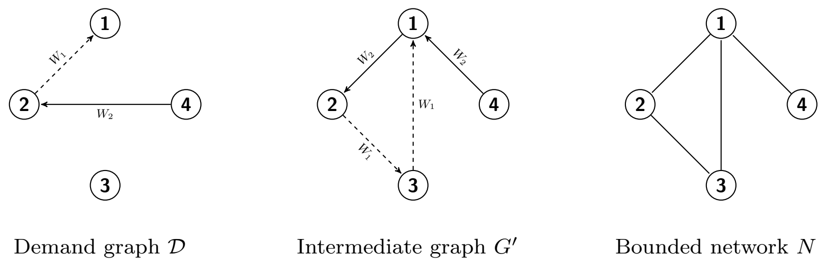

Figure 1: In this example, $\Delta_{\text{avg}} = 1$, thus the limit of the number of times a node can help is $\lceil \Delta_{\text{avg}}/2 \rceil = 1$. Nodes 1 and 3 help the edges $(4, 2)$ and $(2, 1)$, respectively. For each node i, a Mehlhorn tree is built on the set of nodes that helped an edge connected to i. The initial degree of 3 is 0, thus there is no Mehlhorn tree from it. The Nodes 1 and 3 help the only edge of 4 and 1, respectively. Finally, 2 is connected to an outgoing and an incoming edge, thus two Mehlhorn trees including the nodes 1 and 3, will be built from 2.

Note that we skip the set H_i while building the Mehlhorn trees of neighbors of i. However, the connection (possibly indirect) between i and a node $j \in H_i$ appears while building the Mehlhorn tree of j.

We claim that the algorithm has a sufficient number of nodes available as helpers. The total number of available helpers times the number they can help is $n \cdot \beta = \frac{n\Delta_{\text{avg}}}{2} = m$. Thus we have sufficient helpers to help all m edges, and we conclude that the algorithm is well-defined.

Now we upper-bound the maximum final degree of the nodes. A node i is involved in one Mehlhorn tree for each node it helped, in total at most 2β trees. Furthermore, the node i is connected with one edge to Mehlhorn trees G_i^+ and G_i^-. Note that the node i is not involved in the Mehlhorn trees of the intermediate nodes that replaced a node between i and another node. Participation in each Mehlhorn tree adds at most 3 edges to a node, thus its final degree γ_i is

$$\gamma_i \leq 6\beta + 2 \leq 6\left(\frac{\Delta_{\text{avg}}}{2} + 1\right) + 2 = 3\Delta_{\text{avg}} + 8.$$

We conclude that the algorithm produces a network with maximum degree of $3\Delta_{\text{avg}} + 8$.

The improved choice of helper nodes does not influence the route lengths. The optimality of route lengths follows by arguments from [21], and the analysis of near-optimality of Mehlhorn trees follows for our algorithm. □

Finally, we elaborate on the choice of helper nodes. When a node is assigned to help one of its incident edges, we produce the least number of edges in the network. Thus, the best strategy for resource-efficient network design is to assign a node itself to help its own edges first. Still, the analysis holds for arbitrary assignments.

5.2. Round Robin Trees for Sparse Demands

Using Round Robin Trees as ego-trees in the algorithm from Section 5.1 provides approximation guarantees for both route length and congestion.

As we argued in §4.2, the congestion in Round Robin Trees is at most twice the optimal. By the construction of the network for sparse demands, any single node may be present in at most $\beta = \lceil \Delta_{avg}/2 \rceil$ ego-trees (it helps at most β edges). The node may carry the load of all roots of ego-trees, and its load may increase at most β times. This concludes that in the designed network, the congestion is at most $2 \cdot \lceil \Delta_{avg}/2 \rceil \leq \Delta_{avg}$ the optimum congestion.

In comparison, in the algorithm from [21], the guarantee for congestion in the ego-tree built with scheduling via Longest Processing Time First is 4/3. While this is a better guarantee for congestion, this comes at a cost of increased route length: approximation ratio is $4 \cdot \log(12\Delta_{avg})$, where using Round Robin Trees, the approximation ratio is 2. We note that our improved construction for sparse graphs §4.2 itself improves the congestion guarantees in comparison to [21], regardless of the ego-tree used — in our construction, each node may be present in $\Delta_{avg}/2$ ego-trees instead of $2\Delta_{avg}$, which improves the congestion guarantees by a factor of 4.

5.3. Computational Complexity

We next investigate the computational complexity of minimizing congestion. To this end we perform a reduction from the k-Vertex Cover problem [22], namely:

Definition 5 (k-Vertex Cover [22])**.** *Given an undirected graph $G = (V, E)$ and a parameter $k \in \mathbb{N}^+$, find a subset $V' \subseteq V$ with $|V'| = k$, s.t. each edge $e \in E$ is incident to least one node $v \in V'$.*

Theorem 5. *The cong-BND (Definition 2) problem is NP-hard.*

Proof. We prove the claim by a reduction from k-Vertex Cover, which remains NP-complete on 3-regular graphs [22]. W.l.o.g., we can assume that $k \in \mathbb{N}^+$ is an even number. Given an instance $I = (G_U = (U, E_U), k)$ of the k-Vertex Cover problem, where G_U is a 3-regular graph with $|U| = n$ and $|E_U| = m$, we construct an instance $I' = (G_D = (V, E), D, \Delta = k)$ of the cong-BND problem as follows. For each node $u_i \in U$, there is a node $v_i \in V$ in I', and for each edge $\{u_i, u_j\} \in E_U$, there are a node v_{ij} and two edges $\{v_i, v_{ij}\} \in E$ and $\{v_i, v_{ij}\} \in E$, where $D(v_i, v_{ij}) = \alpha$ and $D(v_{ij}, v_j) = \alpha$ for a large constant $\alpha > 0$. Moreover, there is a node $v_0 \in V$, which has a demand of $D\left(v_0, v_{ij}\right) = 1$ for each edge $\{u_i, u_j\} \in E_U$.

Additionally, we introduce a gadget A_i, which is a graph of $k+1$ nodes, where $V(A_i) = V_{A_i} = \{a_{i,0}, a_{i,1}, \ldots, a_{i,k}\}$, s.t. every two distinct nodes $a_{i,l} \in V_{A_i}$ and $a_{i,j} \in V_{A_i}$, where $l, j \in \{1, \ldots, k\}$, have demands: $D(a_{i,j}, a_{i,l}) = (\alpha+1)/2$ and $D(a_{i,l}, a_{i,j}) = (\alpha+1)/2$. Moreover, for each node $a_{i,l} \in V_{A_i}$, where $l \in \{1, \ldots, k-1\}$, we have demands: $D(a_{i,0}, a_{i,l}) = (\alpha+1)/2$ and $D(a_{i,l}, a_{i,0}) = (\alpha+1)/2$.

Now, for each node $v_i \in V$, we construct $(k-4)/2$ gadgets, denoted by $\mathcal{A}_i = \{A^i_1, \ldots, A^i_{(k-4)/2}\}$, s.t., in each gadget $A^i_j \in \mathcal{A}_i$, these two nodes $a^i_{j,0} \in V(A^i_j)$ and $a^i_{j,k} \in V(A^i_j)$ have two demands: $D(a^i_{j,0}, v_i) = (\alpha+1)$ and $D\left(a^i_{j,k}, v_i\right) = (\alpha+1)$. Similarly, for each node $v_{ij} \in V$, we construct $(k-2)/2$ gadgets $\mathcal{A}_{ij} = \{A^{ij}_1, \ldots, A^{ij}_{(k-2)/2}\}$, s.t., in each gadget $A^{ij}_l \in \mathcal{A}_{ij}$, these two nodes $a^{ij}_{l,0} \in V\left(A^{ij}_l\right)$ and $a^{ij}_{l,k} \in V\left(A^{ij}_l\right)$ have two demands: $D\left(a^{ij}_{l,0}, v_{ij}\right) = (\alpha+1)$ and $D\left(a^{ij}_{l,k}, v_{ij}\right) = (\alpha+1)$.

We claim that I' has a host network G_H to serve D with maximum load of $\leq \alpha+1$ iff the graph G_U has a size k vertex cover.

First, we note that if two arbitrary nodes $u, v \in V$ have demands $D(u,v) + D(v,u) \geq \alpha+1$, then $\{u,v\} \in E(G_H)$, otherwise the maximum load cannot satisfy $\alpha+1$. Thus, each gadget in G_D must be preserved in G_H, s.t., each demand $D(u,v)$, where $u, v \in V$ in the gadget implies an edge $\{u,v\}$ in G_H. Thus, to construct the remaining parts of G_H, we can think that each node $v_i \in V$ (resp., $v_{ij} \in V$) has a degree bound $\Delta(v_i) = 4$ (resp., $\Delta(v_{ij} \in V) = 2$), while the node v_0 has a degree bound of $\Delta(v_0) = k$.

If G_U has a vertex cover $C \subseteq V_U$ of size k, then, for each edge $\{u_i, u_j\} \in E_U$, we have $\{v_i, v_{ij}\} \in E(G_H)$ and $\{v_i, v_{ij}\} \in E(G_H)$, and for each node $u_i \in C$, we have $\{v_i, v_0\} \in E(G_H)$. It's easy to note that the maximum load is $\alpha+1$.

Conversely, we assume that G_U cannot have a vertex cover of size $\leq k$. Let $C \subseteq V_U$ be a set of k arbitrary nodes in V_U. First, in G_H, we can always have $\{v_i, v_{ij}\} \in E(G_H)$ and $\{v_i, v_{ij}\} \in E(G_H)$ for each edge $\{u_i, u_j\} \in E_U$ since it has $\Delta(v_{ij}) = 2$ and $\Delta(v_i) = 4$, while G_U is a 3-regular graph. For each node $u_i \in C$, we can have an edge $\{v_i, v_0\} \in E(G_H)$. Let $\{u_{i^*}, u_{j^*}\} \in E_U$ be an edge not covered by C, then we know $\{u_{i^*}, v_0\} \notin E(G_H)$ and $\{u_{j^*}, v_0\} \notin E(G_H)$. However, to serve demand $D(v_0, v_{i^*j^*})$, there must be a path from v_0 to $v_{i^*j^*}$ in G_H, which must pass through at least an edge between $\{v_{ii^*}, v_{i^*}\}$ and $\{v_{jj^*}, v_j\}$, where $\{u_i, u_{i^*}\} \in E_U$ and $\{u_j, u_{j^*}\} \in E_U$. Since $\{v_{ii^*}, v_{i^*}\}$ (resp., $\{v_{j^*i^*}, v_j\}$) already has a load of $\alpha + 1$ before serving $D(v_0, v_{i^*j^*})$, then the maximum load in G_H must be $> \alpha + 1$. $\square$

6. exact degree bounds discussion

In this section, we discuss the applicability of our algorithms for real-world deployments in, e.g., data center settings. Recall that so far, we proposed network design algorithms for sparse networks, where the node degree depends on the average node degree of the demand graph. Our sparse network designs are a good fit for data center networks, as they commonly have uniform constant node degrees for practical reasons: deploying the same hardware across the board makes management and repair easy, and also saves on purchasing costs. Hence, if the average node degree of the demand graph is expected to remain stable, our algorithms implicitly also propose the connectivity of each node w.r.t. hardware deployment.

On the other hand, we might face a situation where 1) the hardware is already deployed, or where 2) the demand matrix is relatively unstable, shifting between low and high connectivity. In both cases, it could be that *a*) the deployed hardware cannot realize our network designs, or that *b*) our algorithms underutilize the network's potential, by deploying fewer links than the network could realize. We next briefly discuss how to take advantage of both situations (§6.1 and §6.2) and then perform a small case study for traffic traces from Facebook's data centers in §6.3. Lastly, we discuss some take-aways of our case study in §6.4.

6.1. Heavy Traffic: High Degree

In case the deployed hardware cannot support the required node degrees from our algorithms, we can no longer guarantee the specified performance bounds on route length and congestion. However, as data center traffic is often skewed [14], we can simply pick the largest demands from the demand graph, until we reach the maximum degree bounds. In this fashion, we retain guarantees for the majority of traffic, but still need to serve the remaining demands.

Here we can take inspiration from hybrid designs [25], which defer some part of the demands to a static topology. For example, by borrowing three reconfigurable ports at each node, we can build a network structure of logarithmic depth, e.g., a balanced binary tree or even better, an expander [26, 27]. In particular for expanders, one could study heuristics that borrow a larger number of reconfigurable ports [4]. However, we are interested in network design with provable performance guarantees and hence leave such investigations to future work.

Even more so, this static network would not need to be reconfigurable, and we propose to hence fall back to well-understood demand-oblivious data center network designs, such as tried and tested (folded) Clos networks [28, 29] or even recent static expander designs [30, 31, 32]. In this fashion, the large majority of demands enjoys the full flexibility of the reconfigurable topology design, whereas the small remainder is routed on a small static topology.

6.2. Light Traffic: Low Degree

When the network's nodes are over-provisioned, we are in a comfortable situation, as we can guarantee all our algorithms' performance bounds. Additionally, if the over-provision is significant, we can utilize additional links to decrease route lengths or decrease congestion. We can scale up to additional degrees by utilizing (α, β)-Round Robin Trees for larger values of α or β. We note that improvements in route lengths and congestion are independent, and we can trade additional links for improvements for either of them.

6.3. Case Study: Facebook's Data Center Traffic

We investigate Facebook's data center traffic data [33, 34, 35], using their database and Hadoop clusters at the pod level. Their data set covers a sample

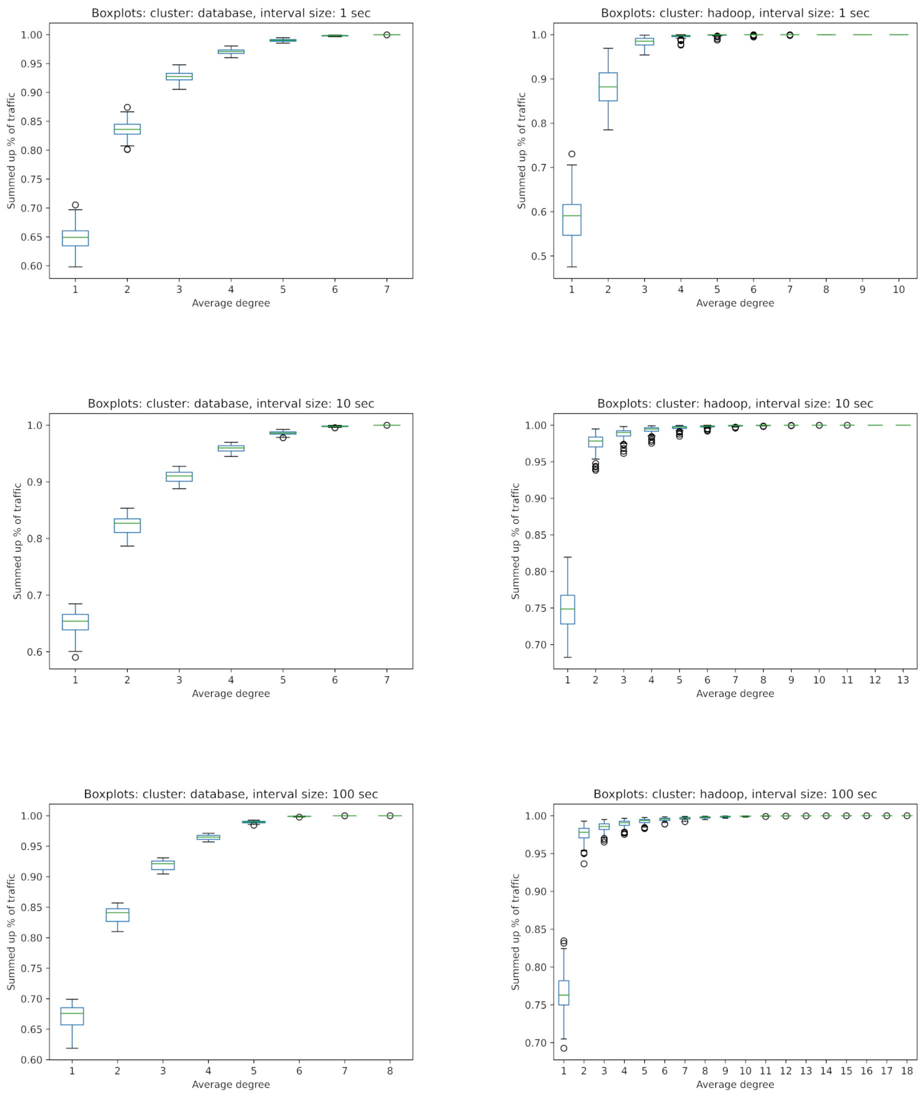

Figure 2: Boxplots showing the % of traffic covered by different average degrees for Facebook's database (left) and hadoop (right) cluster at the pod level, with 110 and 109 nodes, respectively.

of traffic traces over 24 hours. Figure 2 shows boxplots for 100 samples taken uniformly spaced over the course of the day, for interval sizes of 1, 10, and 100 seconds. For each of these samples, we take all the traffic at the pod level and investigate what level of average degree can cover how much of the total traffic.

As can be seen and as expected, the average degree goes up over longer interval sizes, very slightly from 7 to 8 for the database cluster, and from 10 to 18 for the hadoop cluster. On the other hand, with longer interval sizes, the amount of traffic covered by small average degrees increases: from a median of $\approx 65\%$ to $\approx 68\%$ for the database cluster, and even from $\approx 59\%$ to $\approx 76\%$ for the hadoop cluster. While the following average degree sizes remain relatively stable in their traffic coverage for the database cluster, the trend continues for the hadoop cluster: for example, for an average demand degree of 2, from $\approx 88\%$ to $\approx 98\%$.

In terms of hardware feasibility, we consider both free-space optics architectures similar to *ProjecToR* [10] and emerging optical circuit switches. First, *ProjecToR* proposes to use 16 lasers and receivers for 128 nodes in their simulations. For our spare network designs with maximum degree $\Delta_{\max} = 3\Delta_{\text{avg}} + 8$, we can hence utilize an average demand degree of 2, due to $3 \cdot 2 + 6 = 14 < 16$, covering a median of between $\approx 84\%$ to $\approx 98\%$ of the traffic, but at least about 80%. Second, e.g., already Alistarh et al. [36] demonstrate the feasibility of efficient optical switches in the order of 1000 ports, i.e., our designs could cover about two thirds of the traffic. Very recent work [5] moreover showcases how to connect 25,600 ports by means of optical switching. As such, our designs scale to over 2300 nodes for an average demand degree of 1, over 1800 for $\Delta_{\text{avg}} = 2$, over 1500 for $\Delta_{\text{avg}} = 3$, over 1200 for $\Delta_{\text{avg}} = 4$ etc.

6.4. Discussion

The purpose of our evaluation in §6.3 was to investigate the applicability of our designs to a common data set for data center traffic traces, namely by investigating the average (traffic) degree. The average degree of the traffic influences the maximum degree of the constructed network, which influences the viability of the proposed solution, i.e., how it can be deployed in practice.

Our case study showed that a majority of all traffic in the Facebook traffic traces can be covered with a relatively small average degree. Hence, they fit well for our proposed network designs, as our algorithms can in turn construct graph structures with good stretch and congestion guarantees, only requiring small constant degree nodes, as prevalent in data center hardware. In contrast, for traffic with high average degree (e.g., in all-to-all communication patterns), our algorithms would not provide solutions with strict guarantees, respectively only for very small parts of the traffic.

Hence, we believe it would be of interest and promising to perform further evaluations w.r.t. to the actual deployment of our algorithms in a data center context, e.g., by comparison to other network design methods and to moreover investigate how our methods can be best adapted to a dynamic context, where network reconfiguration comes at a cost and upcoming traffic might not able to be predicted perfectly.

7. Related Work

Reconfigurable datacenter networks have received much attention recently [6, 7, 8, 9, 10, 12, 5], and we refer the reader to recent surveys [37, 38] for a detailed overview. The focus of our paper is on the underlying network design problem. While there exist results on non-polynomial time exact algorithms and heuristics, e.g., [39, 40], we are interested in polynomial-time algorithms which come with provable (approximation) guarantees.

The optimization problem considered in this paper is related to classic graph-theoretical problems such as the Minimum Cost Communication Spanning Tree problem. On the one hand, our problem is a subproblem of these problems as we restrict the host topology to have a bounded degree Δ; on the other hand our problem is more general in that the designed graph does not have to be a tree. The Minimum Communication Spanning Tree admits an $O(\log^2 n)$-approximation, where n is the number of nodes in the network [41]. For unit demands among all pairs of vertices, the problem is called the Minimum Routing Cost Spanning tree, and it admits a PTAS [42].

Our problem also features interesting connections to arrangement and virtual network embedding problems. If we restrict the maximum degree of the designed network to 2, the studied problem is a subproblem of Minimum Circular Arrangement (a variant of Minimum Linear Arrangement). The problem is solvable in polynomial time [43], with recent corrections to the technical details of the algorithm [44]. More generally, a natural approach to designing demand-aware networks of bounded degree Δ could be to proceed in two stages: first, choose any "good" graph of degree Δ, e.g., a regular graph of small diameter; and then, simply map the nodes of the demand matrix to this graph. The latter problem is known as the Virtual Network Embedding Problem (VNEP), which is known to be NP-hard [45] but for which there also exist approximation algorithms [46]. The problem is related to VLSI layout problems [47, 48].

Prior algorithmic work on demand-aware network designs often focuses on scenarios where there is only one optical link per node [25, 49, 50], augmenting a demand-oblivious network by means of a disconnected matching. In the context of $b>1$ optical links per node, prior work explores a solution generating b-matchings [51, 52] as a means of topology augmentation. While Chen et al. [9] propose to connect the b-matching designs via edge-exchanges, their algorithms have no route length or congestion guarantees. However in a geometric context (e.g., in sensor networks), demand-aware spanner constructions can be designed which obtain route length guarantees [53].

The work closest to ours is by Avin et al. [20, 12] who investigate demand-aware network designs of bounded degree, providing several interesting approximation algorithms, in particular a constant-approximation for the weighted route length objective for sparse demands. The paper already had several followups, e.g., a robust demand-aware network has been proposed in [54], and a version which also minimizes congestion in [21]. In this paper, we improve upon these results by presenting network designs with significantly lower degree requirements, hence reducing infrastructure costs and improving scalability; in fact the results in [21] do not apply to demand graphs with average degree larger than 1/12 of the maximum degree available in the data center.

8. Conclusions

This paper revisited the design of demand-aware networks minimizing route lengths and congestion based on the traffic pattern. In particular, we presented improved network topologies of significantly lower degrees, making our approach more practical: our designs reduce required infrastructure costs and improve scalability.

We regard our work as a first step and believe that it opens several interesting avenues for future research. In particular, some of our bounds are still not tight and it would be interesting to further explore lower bounds on the achievable approximation ratio for different given degrees. Furthermore, it would be interesting to explore the power of randomized algorithms in this context, as well to investigate additional objective functions, e.g., related to throughput or flow completion times. Moreover, in future work we would like to further investigate our demand-aware network designs in hybrid architectures, combining demand-oblivious and reconfigurable topologies (similarly to ReNet [55]), and to explore designs which come with guarantees over time (like SplayNets [56] or ProjecToR [57]).

Acknowledgements

We would like to thank our shepherd Bhuvan Urgaonkar and the reviewers for their valuable feedback. This research was supported by the European Research Council (ERC), grant agreement No. 864228 (AdjustNet), Horizon 2020, 2020-2025.

References

[1] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh, et al., Hpcc: high precision congestion control, in:

Proceedings of the ACM Special Interest Group on Data Communication, 2019, pp. 44–58.

[2] J. C. Mogul, L. Popa, What we talk about when we talk about cloud network performance, ACM SIGCOMM Computer Communication Review 42 (5) (2012) 44–48.

[3] W. M. Mellette, R. McGuinness, A. Roy, A. Forencich, G. Papen, A. C. Snoeren, G. Porter, Rotornet: A scalable, low-complexity, optical datacenter network, in: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, ACM, 2017, pp. 267–280.

[4] W. M. Mellette, R. Das, Y. Guo, R. McGuinness, A. C. Snoeren, G. Porter, Expanding across time to deliver bandwidth efficiency and low latency, in: 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20), USENIX Association, 2020, pp. 1–18.

[5] H. Ballani, P. Costa, R. Behrendt, D. Cletheroe, I. Haller, K. Jozwik, F. Karinou, S. Lange, K. Shi, B. Thomsen, H. Williams, Sirius: A flat datacenter network with nanosecond optical switching, in: SIGCOMM, ACM, 2020, pp. 782–797.

[6] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, B. Y. Zhao, H. Zheng, Mirror mirror on the ceiling: Flexible wireless links for data centers, ACM SIGCOMM Computer Communication Review 42 (4) (2012) 443–454.

[7] S. Kandula, J. Padhye, P. Bahl, Flyways to de-congest data center networks, Proc. ACM Workshop on Hot Topics in Networks (HotNets).

[8] N. Hamedazimi, Z. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, A. Tanwer, Firefly: A reconfigurable wireless data center fabric using free-space optics, in: ACM SIGCOMM Computer Communication Review, Vol. 44, ACM, 2014, pp. 319–330.

[9] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, Y. Chen, Osa: An optical switching architecture for data center networks with unprecedented flexibility, IEEE/ACM Transactions on Networking (TON) 22 (2) (2014) 498–511.

[10] M. Ghobadi, R. Mahajan, A. Phanishayee, N. R. Devanur, J. Kulkarni, G. Ranade, P. Blanche, H. Rastegarfar, M. Glick, D. C. Kilper, Projector: Agile reconfigurable data center interconnect, in: SIGCOMM, ACM, 2016, pp. 216–229.

[11] W. Dai, K. Foerster, D. Fuchssteiner, S. Schmid, Load-optimization in reconfigurable networks: Algorithms and complexity of flow routing, SIGMETRICS Perform. Evaluation Rev. 48 (3) (2020) 39–44.

[12] C. Avin, K. Mondal, S. Schmid, Demand-aware network designs of bounded degree, Distributed Comput. 33 (3-4) (2020) 311–325.

[13] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, R. Chaiken, The nature of data center traffic: measurements & analysis, in: Proc. 9th ACM SIGCOMM conference on Internet measurement, 2009, pp. 202–208.

[14] C. Avin, M. Ghobadi, C. Griner, S. Schmid, On the complexity of traffic traces and implications, Proc. ACM Meas. Anal. Comput. Syst. 4 (1) (2020) 20:1–20:29.

[15] S. Kassing, A. Valadarsky, G. Shahaf, M. Schapira, A. Singla, Beyond fat-trees without antennae, mirrors, and disco-balls, in: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, ACM, 2017, pp. 281–294.

[16] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, et al., Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network, ACM SIGCOMM computer communication review 45 (4) (2015) 183–197.

[17] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, S. Lu, Bcube: a high performance, server-centric network architecture for modular data centers, ACM SIGCOMM Computer Communication Review 39 (4) (2009) 63–74.

[18] C. Avin, C. Griner, I. Salem, S. Schmid, An online matching model for self-adjusting tor-to-tor networks, arXiv preprint arXiv:2006.11148.

[19] R. Andrade, T. Bonates, M. Campêlo, M. Ferreira, Minimum linear arrangements, Electronic Notes in Discrete Mathematics 62 (2017) 63 – 68, lAGOS'17 – IX Latin and American Algorithms, Graphs and Optimization.

[20] C. Avin, K. Mondal, S. Schmid, Demand-aware network designs of bounded degree, in: Proc. International Symposium on Distributed Computing (DISC), 2017.

[21] C. Avin, K. Mondal, S. Schmid, Demand-aware network design with minimal congestion and route lengths, in: Proc. IEE INFOCOM, 2019.

[22] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, 1979.

[23] R. L. Graham, Bounds on multiprocessing timing anomalies, SIAM Journal of Applied Mathematics 17 (1969) 416–429.

[24] K. Mehlhorn, Nearly optimal binary search trees, Acta Informatica, v.5, 287-295 (1975) 5.

[25] K. Foerster, M. Ghobadi, S. Schmid, Characterizing the algorithmic complexity of reconfigurable data center architectures, in: ANCS, ACM, 2018, pp. 89–96.

[26] M. Ajtai, Recursive construction for 3-regular expanders, Comb. 14 (4) (1994) 379–416.

[27] E. Kowalski, An introduction to expander graphs, Société Mathématique de France, 2019.

[28] C. Clos, A study of non-blocking switching networks, Bell System Technical Journal 32 (2) (1953) 406–424.

[29] M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center network architecture, in: SIGCOMM, ACM, 2008, pp. 63–74.

[30] A. Valadarsky, M. Dinitz, M. Schapira, Xpander: Unveiling the secrets of high-performance datacenters, in: HotNets, ACM, 2015, pp. 16:1–16:7.

[31] A. Valadarsky, G. Shahaf, M. Dinitz, M. Schapira, Xpander: Towards optimal-performance datacenters, in: CoNEXT, ACM, 2016, pp. 205–219.

[32] M. Dinitz, M. Schapira, A. Valadarsky, Explicit expanding expanders, Algorithmica 78 (4) (2017) 1225–1245.

[33] A. Roy, H. Zeng, J. Bagga, G. Porter, A. C. Snoeren, Inside the social network's (datacenter) network, in: SIGCOMM, ACM, 2015.

[34] J. H. Zeng, Data sharing on traffic pattern inside facebook's datacenter network, https://research.fb.com/data-sharing-on-traffic-pattern-inside-facebooks-datacenter-network/ (Jan. 2017).

[35] facebook, Facebook network analytics data sharing, https://www.facebook.com/groups/1144031739005495/ (2018).

[36] D. Alistarh, H. Ballani, P. Costa, A. C. Funnell, J. Benjamin, P. M. Watts, B. Thomsen, A high-radix, low-latency optical switch for data centers, in: SIGCOMM, ACM, 2015, pp. 367–368.

[37] K. Foerster, S. Schmid, Survey of reconfigurable data center networks: Enablers, algorithms, complexity, SIGACT News 50 (2) (2019) 62–79.

[38] M. N. Hall, K. Foerster, S. Schmid, R. Durairajan, A survey of reconfigurable optical networks, Opt. Switch. Netw. 41 (2021) 100621.

[39] A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, Proteus: a topology malleable data center network, in: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, ACM, 2010, p. 8.

[40] M. Wang, Y. Cui, S. Xiao, X. Wang, D. Yang, K. Chen, J. Zhu, Neural network meets dcn: Traffic-driven topology adaptation with deep learning, Proceedings of the ACM on Measurement and Analysis of Computing Systems 2 (2) (2018) 26.

[41] D. Peleg, E. Reshef, Deterministic polylog approximation for minimum communication spanning trees, in: Proc. 22nd Int. Colloq. on Automata, Languages and Programming (ICALP), 1989, pp. 670–681.

[42] B. Y. Wu, G. Lancia, V. Bafna, K.-M. Chao, R. Ravi, C. Y. Tang, A polynomial-time approximation scheme for minimum routing cost spanning trees 29 (3) (2000) 761–778.

[43] Y. Shiloach, A minimum linear arrangement algorithm for undirected trees 8 (1) (1979) 15–32.

[44] J. L. Esteban, R. F. i Cancho, A correction on shiloach's algorithm for minimum linear arrangement of trees 46 (3) (2017) 1146–1151.

[45] M. Rost, S. Schmid, On the hardness and inapproximability of virtual network embeddings, in: IEEE/ACM Transactions on Networking (TON), 2020.

[46] M. Rost, S. Schmid, Virtual network embedding approximations: Leveraging randomized rounding, in: Proc. IEEE/ACM Transactions on Networking (ToN), 2019.

[47] S. N. Bhatt, S. S. Cosmadakis, The complexity of minimizing wire lengths in vlsi layouts, Information Processing Letters 25 (4) (1987) 263–267.

[48] S. N. Bhatt, F. T. Leighton, A framework for solving vlsi graph layout problems, Journal of Computer and System Sciences 28 (2) (1984) 300–343.

[49] T. Fenz, K. Foerster, S. Schmid, A. Villedieu, Efficient non-segregated routing for reconfigurable demand-aware networks, Comput. Commun. 164 (2020) 138–147.

[50] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, A. Vahdat, Helios: a hybrid electrical/optical switch architecture for modular data centers, ACM SIGCOMM Computer Communication Review 41 (4) (2011) 339–350.

[51] K. Foerster, M. Pacut, S. Schmid, On the complexity of non-segregated routing in reconfigurable data center architectures, Computer Communication Review 49 (2) (2019) 2–8.

[52] M. Bienkowski, D. Fuchssteiner, J. Marcinkowski, S. Schmid, Online dynamic b-matching with applications to reconfigurable datacenter networks, in: Proc. 38th International Symposium on Computer Performance, Modeling, Measurements and Evaluation (PERFORMANCE), 2020.

[53] E. Ceylan, K. Foerster, S. Schmid, K. Zaitsava, Demand-aware plane spanners of bounded degree, in: Networking, IEEE, 2021, pp. 1–9.

[54] C. Avin, A. Hercules, A. Loukas, S. Schmid, rdan: Toward robust demand-aware network designs, in: Information Processing Letters (IPL), 2018.

[55] C. Avin, S. Schmid, Renets: Statically-optimal demand-aware networks, in: Proc. SIAM Symposium on Algorithmic Principles of Computer Systems (APOCS), 2021.

[56] S. Schmid, C. Avin, C. Scheideler, M. Borokhovich, B. Haeupler, Z. Lotker, Splaynet: Towards locally self-adjusting networks, IEEE/ACM Trans. Netw. 24 (3) (2016) 1421–1433.

[57] J. Kulkarni, S. Schmid, P. Schmidt, Scheduling opportunistic links in two-tiered reconfigurable datacenters, in: 33rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), 2021.