# Traffic Engineering with Equal-Cost-MultiPath: An Algorithmic Perspective

Marco Chiesa
Department of Engineering
Roma Tre University
chiesa@dia.uniroma3.it

Guy Kindler
School of Computer
Science and Engineering
Hebrew University of Jerusalem
gkindler@cs.huji.ac.il

Michael Schapira
School of Computer
Science and Engineering
Hebrew University of Jerusalem
schapiram@huji.ac.il

***Abstract*—To efficiently exploit network resources operators do traffic engineering (TE), i.e., adapt the routing of traffic to the prevailing demands. TE in large IP networks typically relies on configuring static link weights and splitting traffic between the resulting shortest-paths via the Equal-Cost-MultiPath (ECMP) mechanism. Yet, despite its vast popularity, crucial operational aspects of TE via ECMP are still little-understood from an algorithmic viewpoint. We embark upon a systematic algorithmic study of TE with ECMP. We consider the standard model of TE with ECMP and prove that, in general, even *approximating* the optimal link-weight configuration for ECMP within *any* constant ratio is an intractable feat, settling a long-standing open question. We establish, in contrast, that ECMP can provably achieve optimal traffic flow for the important category of Clos datacenter networks. We last consider a well-documented shortcoming of ECMP: suboptimal routing of large (“elephant”) flows. We present algorithms for scheduling “elephant” flows on top of ECMP (as in, e.g., Hedera [1]) with *provable* approximation guarantees. Our results complement and shed new light on past experimental and empirical studies of the performance of TE with ECMP.**

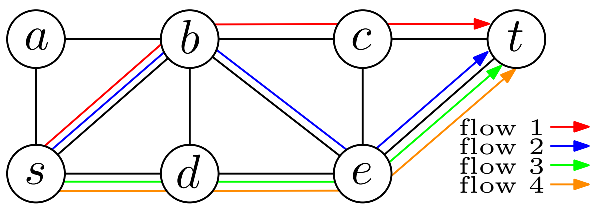

Fig. 1: An illustration of Equal-Cost-MultiPath routing: 4 TCP connections, called “flows 1-4”, originate at (source) router $s$ and are destined for (target) router $t$. All link weights are 1. Observe that $(s, b, c, t)$, $(s, b, e, t)$ and $(s, d, e, t)$ are (all) the induced shortest-paths from $s$ to $t$. Each router now uses a static hash function on packet headers to map every connection to an outgoing link on a shortest-path to its destination, e.g., router $s$ can map each of the flows 1-4 to the link $(s, b)$ or the link $(s, d)$ according to its hash function. The figure describes a possible mapping of flows to outgoing links.

## I. Introduction

The rapid growth of online services (from video streaming to 3D games and virtual worlds) is placing tremendous demands on the underlying networks. To make efficient use of network resources, adapt to network conditions, and satisfy user demands, network operators do traffic engineering (TE), i.e., tune routing-protocol parameters to control how traffic is routed across the network. Our focus in this paper is on the prevalent mechanism for engineering the flow of traffic within a single administrative domain (e.g., company, university campus, Internet Service Provider, and datacenter): TE with Equal-Cost-MultiPath (ECMP) [2] via static link-weight configuration.

Most large IP networks run Interior Gateway Protocols, e.g., Open Shortest Path First (OSPF) [3], to compute all-pairs shortest-paths between routers based on configurable static link weights (where a link’s weight specifies its distance in the shortest-path computation). The ECMP feature was introduced to exploit shortest-path diversity by enabling the “split” of traffic between multiple shortest-paths via per-flow static hashing [4]. See Figure 1 for an illustration of shortest-path routing and ECMP traffic splitting on a simple network topology. Hence, today’s TE often constrains the flow of traffic in two important respects: (1) traffic from a source to a destination in the network can only flow along the shortest paths between them (for the given configuration of link weights); and (2) traffic can only be split between multiple shortest paths (if multiple shortest paths exist) in a very specific manner (as illustrated in Figure 1).

978-1-4799-3360-0/14/$31.00 ©2014 IEEE

Despite many proposals for alternative TE protocols and techniques, “traditional” TE with ECMP remains the prevalent mechanism for engineering the (intradomain) flow of traffic in today’s Internet because, alongside its limitations, TE with ECMP has many advantages over other, more sophisticated schemes: stable and predictable paths, relatively low protocol overhead, implementation in existing hardware, simple configuration language, scalability, a built-in failure recovery mechanism, and more. Still, while ECMP is the subject of much empirical and experimental study (e.g., for ISP networks [5] and for datacenter networks [6]), even crucial operational aspects of TE with ECMP are little-understood from an algorithmic perspective: Can the configuration of link weights be done in a *provably* good manner? What conditions on network topologies lead to desirable TE guarantees? Can algorithmic insights aid in "fixing" ECMP's documented shortcomings, e.g., the suboptimal routing of large ("elephant") flows? We embark on a systematic algorithmic study of TE with ECMP. Our main contributions are discussed below.

**Optimizing link-weight configuration?** In practice, link weight configuration often relies on heuristics, such as setting link weights to be inversely proportional to capacity [7]. While reasonable, these heuristics come with no guarantees. Can link-weight configuration be executed in a *provably* good manner? We consider the standard "splittable-flow model" of TE with ECMP, put forth by Fortz and Thorup [8], [9], [10], and the standard objective of minimizing the maximum link utilization. We settle a long-standing open question by proving a devastating impossibility result: No computationally-efficient algorithm can approximate the optimal link-weight configuration within *any* constant ratio. We show that this inapproximability result extends to other metrics of interest, e.g., maximizing total throughput and minimizing the sum of (exponentially-increasing) link costs (introduced in [8]). Our proof utilizes a new ("graph-power") technique for amplifying an inapproximability factor. We believe that this technique (somewhat inspired by the "diamond graph" in [11]) is of independent interest and may prove useful in other TE (and flow optimization, in general) contexts.

**Optimizing ECMP performance on specific (datacenter) network topologies.** The above negative result establishes that without imposing any restrictions on the network topology, TE with ECMP comes with no reasonable (provable) guarantees whatsoever. What about *specific* network topologies of interest? What conditions on network topology imply good guarantees? We take the first steps in this research direction. We consider two recent proposals for datacenter network topologies: folded Clos networks (VL2 [6]) and hypercubes (BCube [12], MDCube [13],). Our main positive result establishes that in the splittable-flow model, TE with ECMP is optimal for the important category of folded Clos networks. We show, in contrast, that for hypercubes, computing the optimal link weights for ECMP is NP-hard.

Our optimality result for folded Clos networks supports past experimental studied of ECMP in environments with fine-grained traffic splitting. [1] shows that ECMP routing of small ("mice") flows in Clos networks leads to good network performance. To avoid TCP packet reordering, ECMP routing splits traffic across multiple paths at an (IP-)flow-level granularity, that is, packets belonging to the same IP flow traverse the same path. [14] advocates replacing today's ECMP traffic splitting scheme with packet-level traffic splitting (i.e., allowing the "spraying" of packets belonging to the same flow across multiple paths). [14] shows, via extensive simulations, that "ECMP-like" traffic splitting at packet-level granularity leads to significantly better load-balancing of traffic in folded Clos networks. Our optimality result provides a strong theoretical justification for this claim.

**Optimizing the routing of elephant flows.** As explained above, ECMP splits traffic across multiple paths at an (IP-)flow-level granularity. Consequently, a key limitation of ECMP is that large, long-lived ("elephant") flows traversing a router can be mapped to the same output port. Such "collisions" can cause load imbalances across multiple paths and network bottlenecks, resulting in substantial bandwidth losses [14], [1]. Beyond transitioning to ECMP traffic splitting at packet-level, researchers have also examined other possible approaches to alleviating this. Recent studies, e.g., Hedera [1] and DevoFlow [15], call for dynamically scheduling elephant flows in datacenter (folded Clos) networks so as to minimize traffic imbalances (while still routing mice flows with ECMP). We now focus on the unsplittable-flow model, which captures the requirement that all packets in a flow (be it long-lived or short-lived) traverse the same path, and investigate the approximability of elephant flow routing. We show that this task is intractable and devise algorithms for approximating the (unattainable) optimum. We discuss the connections between our algorithmic results and past experimental studies along these lines.

**Organization.** We present the standard ECMP routing model in Section II. Our inapproximability result for optimizing link-weight configuration is presented in Section III. We discuss our results for TE with ECMP for specific (datacenter) network topologies (folded Clos networks and hypercubes) in Section IV. Our results for scheduling elephant flows in folded Clos networks appear in Section V. We conclude and present directions for future research in Section VII. Due to space constraints many proofs are deferred to the full version of the paper [16].

## II. EMCP Routing Model

We now present the standard model of TE with ECMP from [10]. We refer the reader to [9], [8], [10] for a more thorough explanation of the model and its underlying motivations. We shall revisit some of the premises of this model in Section V.

**Network and traffic demands.** The network is modeled as an undirected graph $G = (V, E)$, where each edge $e \in E$ has fixed capacity $c_e$. Vertices in $V$ represent routers and edges (links) in $E$ represent physical communication links between routers. We are given a $|V| \times |V|$ demand matrix $D$ such that, for each pair $s, t \in V$, the entry $D_{st}$ specifies the volume of traffic, in terms of units of flow, that (source) vertex $s$ sends to (target) vertex $t$.

**Flow assignments.** A flow assignment is a mapping $f : V \times V \times E \to \mathbb{R}^+ \setminus \{0\}$. $f(s,t,e)$ represents the amount of flow from source $s$ to target $t$ traversing edge $e$. Let $f_e = \Sigma_{s,t \in V} f(s,t,e)$, that is, $f_e$ denotes the total amount of flow traversing edge $e$ . We restrict out attention (unless stated otherwise) to flow assignments that obey two conventional constraints: (1) flow conservation: $\forall v \in V$, $\forall s,t \in V$ such that $v \neq s$ and $v \neq t$, $\Sigma_{e \in E_v} f(s,t,e) = 0$, where $E_v$ is the set of $v$'s incident edges in $E$; (2) demand satisfaction: for all $s,t \in V$ $\Sigma_{e\in O_s} f(s,t,e) = \Sigma_{e\in O_t} f(s,t,e) = D_{s,t}$. (Observe that in some scenarios a flow satisfying the two above conditions must exceed the capacity of some link, i.e., $f_e > c_e$ for some edge $e$).

**Link-weight configurations and routing.** A link-weight configuration is a mapping from edges to nonnegative "weights" $w : E \rightarrow \mathbb{R}^+ \setminus \{0\}$. Every such link-weight configuration $w$ induces the unique flow assignment that adheres to the following two conditions:

- **Shortest-path routing.** Link weights in $w$ induce shortest paths between all pairs of vertices, where a path's length is simply the sum of its link weights. All units of flow sent from source $s$ to target $t$ must be routes along the resulting shortest-paths between them. We next explain how traffic is split between multiple shortest paths.
- **Equal splitting.** All units of flow traversing a vertex $v$ en route to a given target vertex $t$ are equally split across all of $v$'s outgoing links on shortest-paths from $v$ to target $t$.

**Optimizing link weight configuration.** We study the optimization of link-weight configuration for ECMP routing. We consider 3 optimization goals:

- **MIN-ECMP-CONGESTION (MEC).** A natural and well-studied optimization goal is to minimize the maximum link utilization, that is, to engineer a flow assignment $f$ (via link-weight configuration) so that $max_{e\in E}\frac{f_e}{c_e}$ is minimized.
- **MIN-SUM-COST.** Another optimization goal that has been studied in the context of TE with ECMP is MIN-SUM-COST [8], [9], [10], [17]: minimizing the sum of edge-costs under a given flow $\Sigma_e \phi(\frac{f_e}{c_e})$, where $\phi$ is an exponentially-increasing cost function, e.g., $\phi(x) = 2^x$.
- **MAX-ECMP-FLOW (MEF).** MEF can be regarded as the straightforward generalization of classical max-flow objective to the multiple sources / multiple targets (i.e., multicommodity flow) setting. Here the goal is to send as much traffic through the network while (i) not exceeding the demands in $D$ (i.e., possibly violating "demand satisfaction", as defined above) and (ii) not exceeding the link capacities.

**Approximating the optimum.** While in some scenarios computing the optimal solution with respect to the above optimization goals is tractable, in other scenarios this task is NP-hard. We therefore also explore the *approximability* of these goals. We use the following standard terminology. Let $\mathcal{A}$ be an algorithm for a minimization problem $P$. For every instance $I$ of $P$, let $\mathcal{A}(I)$ denote the value of $\mathcal{A}$'s outcome for $I$ and $OPT(I)$ denote the value of the optimal solution for $I$. $\mathcal{A}$ is a polynomial-time $\alpha$-*approximation* algorithm for $P$ for $\alpha \geq 1$ if $\mathcal{A}$ runs in polynomial time and for any instance $I$ of $P$, $\mathcal{A}(I) \leq \alpha{\cdot}OPT(I)$. Similarly an algorithm $\mathcal{A}$ is a polynomial-time $\alpha$-*approximation* algorithm for a maximization problem $P$, for $\alpha \geq 1$, if $\mathcal{A}$ runs in polynomial time and, for any instance $I$ of $P$, $\mathcal{A}(I) \geq \frac{OPT(I)}{\alpha}$.

## III. TE WITH ECMP IS INAPPROXIMABLE!

We settle a long-standing question by showing that optimizing link-weight configuration for ECMP is not only NP-hard but cannot, in fact, be approximated within any "reasonable" factor (unless P=NP) with respect to all 3 optimization goals discussed in Section II: MIN-ECMP-CONGESTION, MIN-SUM-COST, and MAX-ECMP-FLOW. Remarkably, these inapproximability results hold even when the demand matrix has a single nonzero entry, i.e., when only a single router aims to send traffic to another router. Hence, in general, configuring link-weights for ECMP cannot be done in a provably good manner.

*Theorem 3.1:* No computationally-efficient algorithm can approximate the optimum with respect to MIN-ECMP-CONGESTION, MIN-SUM-COST or MAX-ECMP-FLOW, within any constant factor $\alpha \geq 1$ unless $P = NP$, even when the demand matrix has a single nonzero entry.

The remainder of the section provides an overview of the main ideas in the proof of Theorem 3.1 for the MIN-ECMP-CONGESTION and MAX-ECMP-FLOW objectives. The proof for MIN-SUM-COST is more involved and is deferred to the full version of the paper [16].

We henceforth focus on the scenario that the demand matrix has a single nonzero entry. Below, we discuss the three main ingredients of the proof of Theorem 3.1: (1) a new graph-theoretic problem called "MAX-ECMP-DAG", which we prove is inapproximable within a small constant factor; (2) amplifying this inapproximability result for MAX-ECMP-DAG via a new technique to establish that MAX-ECMP-DAG is not approximable within *any* constant factor; and (3) showing that our inapproximability result for MAX-ECMP-DAG implies similar results for both MIN-ECMP-CONGESTION and MAX-ECMP-FLOW.

### *A.* MAX-ECMP-DAG

**MAX-ECMP-DAG.** We present the following graph-theoretic problem called "MAX-ECMP-DAG". In MAX-ECMP-DAG, the input is a capacitated directed acyclic graph (DAG) $H$ and a single source-target pair of vertices $(s,t)$ in $H$. We associate with every sub-DAG $\bar{H}$ of $H$ that contains $s$ and $t$ a flow assignment $f_{\bar{H}}$ as follows. Given $\bar{H}$, the flow assignment $f_{\bar{H}}$ is the max-flow from $s$ to $t$ in $\bar{H}$ subject to the constraint that every vertex in $\bar{H}$ split outgoing flow equally between all of its outgoing edges in $\bar{H}$. The objective in MAX-ECMP-DAG is to find the sub-DAG of $H$ for which the induced flow is maximized, i.e., $\max_{\bar{H}} |f_{\bar{H}}|$.

**Inapproximability result for MAX-ECMP-DAG.** We prove that MAX-ECMP-DAG is inapproximable within a (small) constant factor via a reduction from a hardness result for MIN-ECMP-CONGESTION in [10]. We shall later amplify this inapproximability ratio.

*Theorem 3.2:* Given a MAX-ECMP-DAG instance $I$, distinguishing between the following two scenarios in NP-hard:

- $OPT(I) = 1$
- $OPT(I) = \frac{2}{3}$

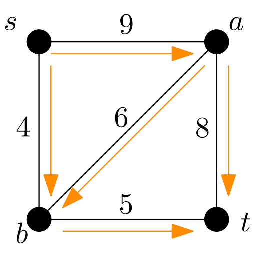

Fig. 2: Graph $G_0$.

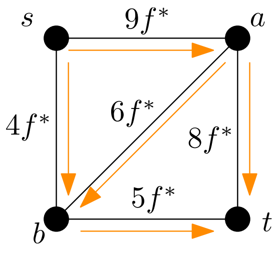

Fig. 3: Abstraction of $G_1$.

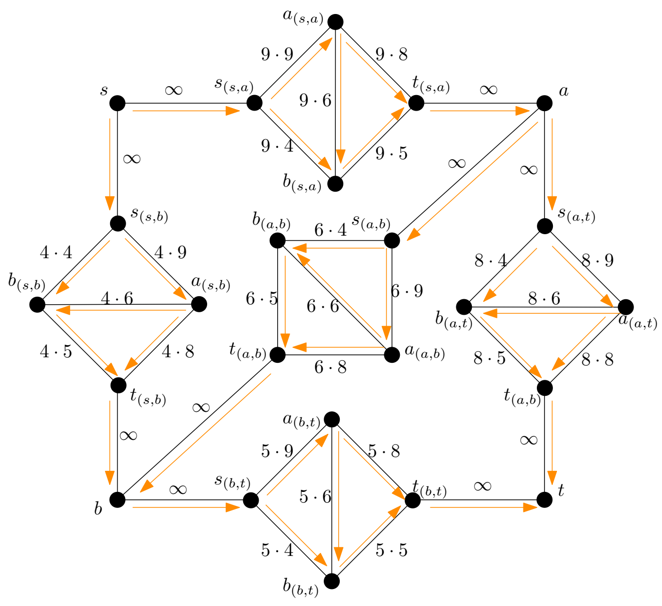

Fig. 4: Graph $G_1$.

where $OPT(I)$ is the value of the optimal solution for $I$.

Observe that Theorem 3.2 implies that MAX-ECMP-DAG cannot be approximated within a factor of $\frac{3}{2}$ (unless P=NP).

## B. Amplifying the Inapproximability Gap

We can now leverage Theorem 3.2 to prove that that MAX-ECMP-DAG is not approximable within any constant factor.

**Amplifying the inapproximability gap: a new technique.** Our proof relies on a new technique for amplifying an inapproximability gap. Roughly speaking, we show how to create, given an instance $I_0$ of MAX-ECMP-DAG, a new, polynomially-bigger, instance $I_1$ of MAX-ECMP-DAG such that $OPT(I_1) = (OPT(I_0))^2$. Observe that as distinguishing between the scenario that $OPT(I_0) = 1$ and the scenario that $OPT(I_0) = \frac{2}{3}$ is NP-hard, distinguishing between the scenario that $OPT(I_1) = 1$ and the scenario that $OPT(I_1) = (\frac{2}{3})^2$ is also NP-hard. By applying this idea multiple times the inapproximability gap can be further amplified to an arbitrary (constant) factor.

**The $\otimes$ operator: intuition.** We now sketch the key tool used in our proof technique. We define the "$\otimes$ operator" that, given two MAX-ECMP-DAG instances, constructs a new MAX-ECMP-DAG instance. Before formally defining the $\otimes$ operator, we illustrate its use via the example in Figure 2. Consider the MAX-ECMP-DAG instance $I_0$ in Figure 2. The numbers in black are edge capacities and the orange arrows indicate the direction of the edges. Observe that the optimal solution for $I_0$ is the sub-DAG that contains the edges $(s, a), (a, b), (b, t)$, and $(a, t)$ and that the value of this solution is 9. Specifically, the optimal solution routes 9 units of flow through $(s, a)$, which are then equally split between$(a, b)$ and $(a, t)$, and the 4.5 units of flow entering vertex $b$ are then sent directly to $t$. Now, consider the instance $I_1$ of MAX-ECMP-DAG, shown in Fig. 4, that is obtained from $I_0$ as follows. Let $G_0$ be the network graph in $I_0$. We replace each edge $(u, v)$ in $G_0$ with an exact copy of $G_0$. We connect vertex $u$ to the source vertex in this copy of $G_0$ and vertex $v$ to the target vertex. The capacity of each edge in this copy of $G_0$ is set to be its original capacity in $G_0$ multiplied by the capacity of $(u, v)$. The capacities of the edges connecting vertices $u$ and $v$ to this copy of $G_0$ are set to be $\infty$.

We argue that the optimal solution for $I_1$, $OPT(I_1)$ is $f^* = 9^2 = 81$. We now provide some intuition for this claim. Let $G_1$ be the network graph in $I_1$. Consider $G_{(u,v)}$, the copy of $G_0$ that was used in the construction of $I_1$ to replace the edge $(u, v)$ in $G_0$. Specifically, consider $G_{(a,b)}$, with $V(G_{(a,b)} = \{s_{a,b}, a_{a,b}, b_{a,b}, t_{a,b}, \}$ and $E(G_{(a,b)}) = \{(s_{(a,b)}, a_{(a,b)}), (s_{(a,b)}, b_{(a,b)}), (a_{(a,b)}, b_{(a,b)}), (a_{(a,b)}, t_{(a,b)}), (b_{(a,b)}, t_{(a,b)})\}$. Observe that the optimal sub-DAG of $G_{(a,b)}$ in terms of maximizing the flow from $s_{(a,b)}$ to $t_{(a,b)}$ is precisely as in the optimal solution for $I_0$. Observe also that the value of the optimal solution within $G_{(a,b)}$ is $9 \times 6$, that is, $f^*$ multiplied by the capacity of the edge $(a, b)$ in $G_0$. Similarly, every subgraph $G_{(u,v)}$ can route a flow of $f^* \times c_{G_0}((u, v))$, where $c_{G_0}((u, v))$ is the capacity of the edge $(u, v)$ in $G_0$. Hence, the network graph $G_1$ can be abstracted as in Figure 3 (replacing each copy of $G_0$ by a single edge with the appropriate capacity). A simple argument shows that the optimal solution in this instance of MAX-ECMP-DAG has value $(f^*)^2$, the value of the optimal solution in $I_0$ multiplied by a scaling factor of $f^*$.

**The $\otimes$ operator: formal definition.** Let $I_1$ and $I_2$ be two MAX-ECMP-DAG instances. We now define the operation $I_1 \otimes I_2$. Let $G_1$ and $G_2$ be the network graphs in $I_1$ and $I_2$, respectively. $I = I_1 \otimes I_2$ is an instance of MAX-ECMP-DAG with network graph $G$ constructed as follows. We create, for every edge $e \in E(G_1)$, a copy of $G_2$, $G_e$. Let $s_e$ and $t_e$ denote the source and target vertices in $G_e$, respectively. The set of vertices in $G$ consists of the vertices in $V(G_1)$ and also of the vertices in all $V(G_e)$'s, i.e., $V(G) = V(G_1) \bigcup_{e \in E(G_1)} V(G_e)$. The set of edges in $G$ contains all the edges in the different $E(G_e)$'s, and also the edges $(u, s_e)$ and $(t_e, v)$ for every edge $e = (u, v) \in E(G_1)$, i.e., $E(G) = \bigcup_{e=(u,v) \in E(G_1)} (\{(u, s_e)(t_e, v)\} \cup E(G_e))$. The capacity of every edge in $G_e$ is set to be the capacity of the corresponding edge in $G_2$ multiplied by the capacity of $e$ in $I_1$. The capacity of every edge of the form $(u, s_e)$ or $(t_e, v)$ is set to $\infty$.

**Gap amplification via the $\otimes$ operator.** We prove a crucial property of the $\otimes$ operator: applying the $\otimes$ operator to an instance $I$ of MAX-ECMP-DAG $k$ times increases the value of the optimal solution from $OPT(I)$ in the original instance $I$ to $(OPT(I))^k$ in the resulting new instance of MAX-ECMP-DAG.

*Lemma 3.3:* Let $I$ be an instance of MAX-ECMP-DAG. $OPT(\otimes^k I) = (OPT(I))^k$ for any integer $k > 0$.

Lemma 3.3 can now be used to prove that no constant approximation ratio is achievable for MAX-ECMP-DAG. Recall that, by Theorem 3.2, distinguishing, for a given a MAX-ECMP-DAG instance $I$, between the following two scenarios in NP-hard: (1) $OPT(I) = 1$; and (2) $OPT(I) = \frac{2}{3}$. Observe that when combined with Lemma 3.3 this implies that distinguishing, for a given a MAX-ECMP-DAG instance $I$, between the following two scenarios is also NP-hard: (1) $OPT(I) = 1$; and (2) $OPT(I) = (\frac{2}{3})^k$ for any constant integer $k > 0$.

## *C. Relating* MAX-ECMP-DAG *to* MIN-ECMP-CONGESTION *and* MAX-ECMP-FLOW

We present the following lemma, which concludes the proof.

*Lemma 3.4:* For any $\alpha > 1$, if MAX-ECMP-DAG is NP-hard to approximate within a factor of $\alpha$ then

- MIN-ECMP-CONGESTION is NP-hard to approximate within a factor of $\alpha$ in the single source-target pair setting;
- MAX-ECMP-FLOW is NP-hard to approximate within a factor of $\alpha$ in the single source-target pair setting.

Our proof of this lemma also involves the application of the $\otimes$ operator and is deferred to the full version of this paper [16].

## IV. TE WITH ECMP IN DATACENTER NETWORKS

We now explore the guarantees of TE with ECMP in two specific network topologies, which have recently been studied in the context of datacenter networks: folded Clos networks and hypercubes. We prove that while in hypercubes optimal TE with ECMP remains intractable, ECMP routing easily achieves the optimal TE outcome in folded Clos networks. Our positive result for folded Clos networks implies that TE with ECMP is remarkably good when traffic consists of a large number of small (mice) flows (see Hedera [1]), or when traffic is split at a packet-level (instead of IP-flow-level, e.g., via Random Packet Spraying [14]), as in these contexts the splittable-flow model well-captures the network behavior. We discuss the handling of unsplittable large (elephant) flows in Section V.

### *A. TE with ECMP is Optimal for Folded Clos Networks*

We now present our optimality result for TE with ECMP in folded Clos networks (FCNs).

**Folded Clos networks.** An n-FCN is a graph whose vertices are partitioned into $n$ sets, called stages, that is obtained via the following recursive construction:

- **A 1-FCN.** A 1-FCN consists of a single stage ("stage 1") that contains a single vertex.
- **Construction of an n-FCN from an (n-1)-FCN.** Let $F^{n-1}$ be an (n-1)-FCN. An n-FCN $F^n$ is constructed as follows:
  - **Creating stages $1,\dots,n-1$ of $F^n$:** Create, for some chosen $k > 0$, $k$ duplicates of $F^{n-1}$: $F_1^{n-1},\dots,F_k^{n-1}$. Set stage $i = 1,\dots,n-1$ of $F^n$ to be the union of the $i$'th stages of $F_1^{n-1},\dots,F_k^{n-1}$. Create an edge between two vertices in stages $1,\dots,n-1$ of $F^n$ iff the two vertices belong to the same $F_t^{n-1}$ and there is an edge between the two vertices in $F_t^{n-1}$.
  - **Creating stage $n$ of $F^n$:** Create, for a chosen $r > 0$, $r$ new vertices $v_{i,1},\dots,v_{i,r}$ for every vertex $i$ in the $n-1$'th stage of $F^{n-1}$. Set the $n$'th stage of $F^n$ to be the union $\bigcup_i \{v_{i,1}\dots,v_{i,r}\}$. Create, for every vertex $i$ in the $n-1$'th stage of $F^{n-1}$ an edge between each of the $k$ vertices in the $n-1$'th stage of $F^n$ that correspond to vertex $i$ and each of the vertices in $\{v_{i,1},\dots,v_{i,r}\}$.

Figure 5 shows a 3-FCN constructed by interconnecting six 2-FCNs. Past work focused on the scenario that all link capacities in an FCN are equal (as in [18], [6], [19]). Our positive result below extends to the scenario that only links in the same "layer", that is, that all links that connect the same two stages in the FCN, must have equal capacity.

**TE with ECMP is optimal for Clos networks even when all link weights are 1.** We investigate the complexity of MIN-ECMP-CONGESTION, MIN-SUM-COST, and MAX-ECMP-FLOW, for FCNs. We call a demand matrix for an FCN "inter-leaf" if the sources and targets of traffic are all vertices in stage 1 of the FCN (i.e., the leaves of the multi-rooted tree). Inter-leaf demand matrices capture realistic traffic patterns in datacenters, as most traffic in a datacenter flows between the top-of-rack switches at the lowest level of the datacenter topology. We present a surprising positive result: Setting all links weights to be 1 (i.e, the default in datacenters) results in the optimum traffic flow for *a*ny inter-leaf demand matrix for all three optimization objectives.

*Theorem 4.1:* When all link weights in an FCN network are 1 ECMP routing achieves the optimum flow with respect to MIN-ECMP-CONGESTION, MIN-SUM-COST, and MAX-ECMP-FLOW.

We now prove this result with respect to MIN-ECMP-CONGESTION, and for the scenario that all edge capacities are equal. We defer the proofs for MIN-SUM-COST and MAX-ECMP-FLOW, and also the extension to more general edge capacities, to the full version of the paper [16].

*Proof:* Let $F$ be an $n$-FCN network such that $n \geq 2$ and all link weights are 1. An *l-sub-FCN* of $F$, for $1 \leq l \leq n$ is the subgraph of $F$ that is induced by all vertices in stages $1,\dots,l$ (i.e., the graph consisting of these vertices and edges between them only).

Now, let $S$ be any sub-FCN of $F$ with $l \leq n$ stages of $F$ and let $F_1^{l-1},\dots,F_m^{l-1}$ be all the $(l-1)$-sub-FCNs of $S$ that used in the recursive construction of $S$ (see above). $\bar{V}(S)$ denotes the set of vertices in the last stage of $S$ and $\bar{V}(F_i^{l-1})$, with $i = 1,\dots,m$ denotes the set of vertices in the last stage of $F_i^{l-1}$.

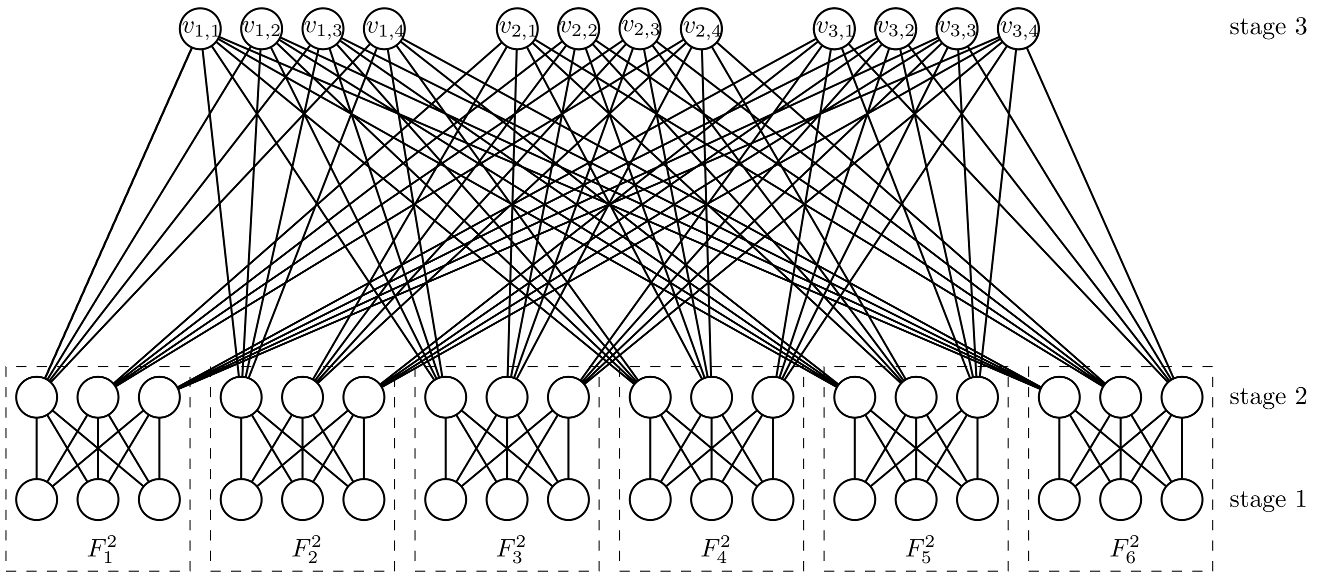

Fig. 5: A 3-FCN constructed by interconnecting four 2-FCNs.

The following claims easily follow from the construction of $F$ and $S$.

*Claim 1:* If $l > 1$ ($S$ has more than one stage), then for every two vertices $v \in \bar{V}(S)$ and $u \in \bar{V}(F_i^{l-1})$ for $i = 1, \ldots, m$, $(u, v)$ is on a shortest-path from $v$ to any vertex in the first stage of $V(F_i^{l-1})$.

*Proof:* We prove by induction on $l$ that the length of the shortest-path from $v$ to any vertex $z$ in the first stage of $F_i^{l-1}$ is $|l-1|$. Clearly, if $l = 2$, then there is a unique path of length 1 between $v$ and every vertex in the first stage. If $l > 2$, then by the induction hypothesis there exists a shortest-path of length $l-2$ from any vertex in $\bar{V}(F_i^{l-1})$ to any vertex $z$ in the first stage of $F_i^{l-1}$. As $v$ is directly connected to a vertex in $\bar{V}(F_i^{l-1})$, and every path to $z$ must cross a vertex in $\bar{V}(F_i^{l-1})$, the claim follows. ∎

*Claim 2:* If $l > 1$ ($S$ has more than one stage), then for every two vertices $v \in \bar{V}(S)$ and $u \in \bar{V}(F_i^{l-1})$ for $i = 1, \ldots, m$, $(u, v)$ is on a shortest-path from $v$ to any vertex in the first stage of $F$ that is not in $V(F_i^{l-1})$.

*Proof:* We prove by induction on $j = n-l$ that the length of the shortest-path from any $u \in \bar{V}(F_i^l)$ to any vertex $z$ in the first stage of $F$ that is not in $V(F_i^{l-1})$ is the same. Observe that if $j = 0$, then, by Claim 1, the shortest path between a vertex in $\bar{V}(S)$ and a vertex $z$ in the first stage of $F$ that is not in $V(F_i^{l-1})$ is $n-1$. As every vertex $u \in \bar{V}(F_i^l)$ is directly connected to a vertex in $\bar{V}(S)$, and as all shortest-paths from $y$ must cross a vertex in $\bar{V}(S)$, the claim follows. Now, if $j > 1$, then by induction hypothesis and by Claim 1, from every vertex in $\bar{V}(S)$ there exists a shortest-path to $z$ (with nonnegative length). Since every vertex $u \in \bar{V}(F_i^l)$ is directly connected to a vertex in $\bar{V}(S)$ and every shortest-path from $u$ must cross a vertex in $\bar{V}(S)$, the claim again follows. ∎

Let $\mathcal{F}_S$ be the set of flows such that (i) the source vertex is in $S$ and the target vertex is not in $S$; or (ii) the sources vertex is in $F_i^l$ for some $i = 1, \ldots, m$ and the target vertex is in some $F_j^l$ for $j \neq i$.

*Claim 3:* Each vertex in $\bar{V}(S)$ receives an equal fraction of every flow $f \in \mathcal{F}_S$.

*Proof:* We prove the claim by induction on $l$, that is, the number of stages of $S$. When $l = 1$, $S$ is simply a 1-FCN and the claim trivially follows. Now, suppose that $l > 1$. By the induction hypothesis, each vertex $v \in \bar{V}(F_i^{l-1})$ receives the same fraction of any flow $f \in \mathcal{F}_S$ whose source is contained in $V(F_i^{l-1})$. Since every vertex in $\bar{V}(F_i^{l-1})$ is connected to the same number of vertices in $\bar{V}(S)$, each vertex $v \in \bar{V}(S)$ must be (directly) connected to precisely one vertex $m_v \in \bar{V}(F_i^{l-1})$. By Claim 2, $v$ is contained in a shortest-path from $m_v$ to the target vertex of $f$, and so each vertex in $\bar{V}(S)$ receives an equal fraction of $f$. ∎

Let $\bar{\mathcal{F}}_S$ be the set of flows such that the target vertex is in $S$ and the source vertex is not in $S$.

*Claim 4:* Each vertex in $\bar{V}(S)$ receives an equal fraction of every flow $\bar{\mathcal{F}}_S$.

*Proof:* We prove the claim by induction on the number of stages $l = n, \ldots, 1$ of $F$. When $l = n$, $\bar{\mathcal{F}}_S = \oslash$ and the statement holds. Otherwise, if $l < n$, let $T$ be a $(l+1)$-sub-FCN of $F$ that contains $S$ as a subgraph. Consider any flow $f \in \bar{\mathcal{F}}_S$. If the source vertex of $f$ is in (not in) $T$, then, by Claim 3 (by the induction hypothesis), each vertex in $\bar{V}(T)$ receives an equal fraction of every flow $f \in \bar{\mathcal{F}}_S$. Since each vertex in $v \in \bar{V}(T)$ is connected to exactly one vertex in $\bar{V}(S)$, each vertex $m_v \in \bar{V}(S)$ is connected to the same number of vertices in $\bar{V}(T)$, and, by Lemma 1, $m_v$ is contained in a shortest path from $v$ to the target vertex of $f$, we have that each vertex in $\bar{V}(S)$ receives an equal fraction of $f$. ∎

Let $E_S$ be the set of edges between vertices in $\bar{V}(S)$ and vertices in stage $l-1$ of $F$. Observe that, by the definition of FCN, the set of vertices in $\bar{V}(S)$ is a vertex-cut of $F$ for all pairs in $\mathcal{F}_S$. Hence, each flow in $\mathcal{F}_S$ and $\bar{\mathcal{F}}_S$ must traverse at least one vertex in $\bar{V}(S)$ and through at least one edge in $E_S$. Let $\mathcal{F}_S^*$ be the sum of all the flows in $\mathcal{F}_S$ and in $\bar{\mathcal{F}}_S$. We have that $\frac{\mathcal{F}_S^*}{c_l|E_S|}$, where $c_l$ is the capacity of edges between vertices in the $l$'th and in the $(l-1)$'th stages, is a lower bound on the amount of flow that is routed through the most loaded edge in $E_S$. We will now prove that when all link weights are 1, this lower bound is achieved (and the theorem follows).

Edges in $E_S$ connect vertices in $\bar{V}(S)$ to vertices in stage $l-1$ of $S$. Since each vertex in $\bar{V}(S)$ is connected to the same number of vertices in stage $l-1$ of $S$ and each vertex in stage $l-1$ of $S$ is connected to the same number of vertices in $\bar{V}(S)$, Claim 3 and Claim 4 imply that each edge carries an equal fraction of each flow in $\mathcal{F}_S^*$. ■

## B. TE with ECMP is NP-hard for Hypercubes

We now investigate MIN-ECMP-CONGESTION in hypercubes. We show that, in contrast to folded Clos networks, MIN-ECMP-CONGESTION in hypercubes is NP-hard.

**Hypercubes.** A *k-hypercube* is a graph in which the set of vertices is $\{0,1\}^k$ and an edge between two vertices $u = (u_1,\ldots,u_k)$ and $v = (v_1,\ldots,v_n)$ exists iff the Hamming distance between $u$ and $v$ is 1 (that is, the two vertices differ in just a single coordinate).

**Optimizing TE with ECMP is intractable for hypercubes.** We present the following hardness result for hypercubes.

*Theorem 4.2:* Computing the optimal flow with respect to MIN-ECMP-CONGESTION in hypercubes is NP-hard.

## V. Routing Elephants in Datacenter Networks

A key shortcoming of ECMP is that large, long-lived ("elephant") flows traversing a router can be mapped to the same output port. Such "collisions" can cause load imbalances across multiple paths and network bottlenecks, resulting in substantial bandwidth losses. To remedy this situation, recent studies, e.g., Hedera [1] and DevoFlow [15], call for dynamically scheduling elephant flows in folded Clos datacenter networks so as to minimize traffic imbalances (while still routing small, "mice" flows via link-state routing and ECMP). We therefore next focus on the so called "unsplittable-flow model".

**Min-Congestion-Unsplittable-Flow (MCUF).** We study the Min-Congestion-Unsplittable-Flow (MCUF) objective: The input is a capacitated graph $G = (V, E, c)$ and a set $\bar{D}$ of "flow demands" of the form $(s, t, \gamma)$ for $s, t \in V$ and $\gamma > 0$, where a single source-target pair $(s, t)$ can appear in more than one flow demand. The goal is to select, for every flow demand $(s, t, \gamma)$, a single shortest-path from $s$ to $t$, such that the maximum load, i.e., $max_e \frac{f_e}{c_e}$, is minimized (as in MIN-ECMP-CONGESTION, see Section II for formal definitions of flow assignments and load). We aim to understand how well unsplittable flows can be routed in datacenter network topologies and, specifically, in FCNs.

**MCUF cannot be approximated within a factor better than 2 even in 2-FCNs.** We show that approximating MCUF within a factor better than 2 is NP-hard even in a 2-FCN, i.e., in a complete bipartite graph. Our proof relies on a reduction from the well-studied (NP-hard) 3-EDGE-COLORING problem [20].

*Theorem 5.1:* Approximating MCUF within a factor of $2-\epsilon$ is NP-hard for 2-FCNs for any constant $\epsilon > 0$.

**A 5-approximation algorithm for 3-FCNs.** We now consider 3-FCNs, which are of much interest in the datacenters context. [18] and VL2 [6] advocate 3-FCNs as a datacenter topology, and Hedera [1] and DevoFlow [15] study the routing of elephant flows in such networks. We present a natural, greedy algorithm for MCUF, called EQUILIBRIUM-ALGO:

- Start with an arbitrary assignment of a single shortest-path for every source-target pair $(s, t)$.
- While there exists a source-destination pair $(s, t)$ such that rerouting the flow from $s$ to $t$ to a different path can either (1) result in a lower maximum load or (2) lower the number of links in the network with the highest load, reroute the flow from $s$ to $t$ accordingly. We call this a "reroute operation".

We show that EQUILIBRIUM-ALGO has provable guarantees. Recall that $D$ is a set of flow demands

*Theorem 5.2:* After $|\bar{D}|$ reroute operations, EQUILIBRIUM-ALGO approximates MCUF in 3-FCNs within a factor of 5.

Theorem 5.1 establishes that even in 2-FCNs (and hence also in 3-FCNs) no approximation ratio better than 2 is achievable. We leave open the question of closing the gap between the lower bound of 2 and upper bound of 5 (see Section VII). We do show that the analysis of EQUILIBRIUM-ALGO is tight for equal-size flows (proof omitted). We point out that the key idea behind EQUILIBRIUM-ALGO (rerouting flows to least loaded paths until reaching an equilibrium) resembles the simulated annealing procedure in Hedera [1] and can be regarded as a first step towards analyzing the provable guarantees of this family of heuristics.

**Proof of 5-approximation guarantee.** We introduce the following notation. Consider a 3-FCN $F$ that contains $k_r$ 2-FCNs, each with $k_b$ vertices in its first stage and $k_m$ vertices in its last stage. Every $i$'th vertex in the last stage of a 2-FCN is connected to the same $k_t$ vertices in the last stage of $F$. Hence, there are $k_t k_m$ vertices in the last stage of $F$. We denote by $b_i^j$ ($m_i^j$) the $i$'th vertex in the first (second) stage of the $j$'th FCN. Each vertex $m_i^j$ is connected to vertices $t_1^j, \ldots, t_{k_t}^j$ in the last stage of $F$. Consider a flow assignment computed by EQUILIBRIUM-ALGO. A flow demand $d \in \bar{D}$ from vertex $s$ to vertex $t$ of size $\gamma_d$ is denoted by $((x, y), \gamma_d)$. For each demand $d \in \bar{D}$, let $p_d$ be the simple path along which $d$ is routed and $c(p_d)$ be the value of the most congested link of $p_d$.

*Lemma 5.3:* Let $d \in \bar{D}$ be a flow demand such that $c(p_d) \geq 5 \cdot OPT$. There exists a path $p'$ between $s$ and $t$ such that $c(p') \leq 5 \cdot OPT - \gamma_d$.

*Proof:* Suppose, by contradiction, that such a path $p'$ does not exists. Let $b_i^j$ and $b_g^l$, with $i, g \in [k_b]$ and $j, l \in [k_r]$, where $[n] = \{1, \ldots, n\}$, be the source and target vertices of $d$, respectively. Observe that, since $OPT \geq \gamma_d$, we have $c(p_d) \geq 5 \cdot OPT \geq 5\gamma_d$. Let $n_b$ ($n'_b$) be the number of edges incident to $b_i^j$ plus the number of edges incident to $b_g^l$ that have congestion at least $5 \cdot OPT$ (at least $5 \cdot OPT - \gamma_d$ and at most $5 \cdot OPT$). We denote by $\mathcal{F}_v$ the amount of flow demands that have $v$ as a source or target vertex. We have that, $\mathcal{F}_{b_i^j} + \mathcal{F}_{b_g^l} \geq n_b(5 \cdot OPT) + n'_b(5 \cdot OPT - \gamma_d) \geq 5n_bOPT + n'_b(5 \cdot OPT - OPT) = 5n_bOPT + 4n'_bOPT$. Let $\mathcal{F}_* = \max\{f^{b_i^j}, f^{b_g^l}\}$. We have $2\mathcal{F}_* \geq 5n_bOPT + 4n'_bOPT$. Since $\mathcal{F}_*$ must necessarily be split among $k_m$ edges, we have $OPT \geq \frac{\mathcal{F}_*}{k_m}$. Combining this bound with the previous one, we obtain $k_mOPT \geq \frac{5n_bOPT + 4n'_bOPT}{2} \Rightarrow k_m \geq \frac{5n_b + 4n'_b}{2} \Rightarrow \frac{k_m}{2} \geq \frac{5}{4}n_b + n'_b$.

Now, let $H$ be the set of indices $h \in [k_m]$ such that both $(b_i^j, m_h^j)$ and $(b_g^l, m_h^l)$ have congestion lower than or equal to $5 \cdot OPT - \gamma_d$. By the above computation, we have that $|H| \geq k_m - n_b - n'_b \geq k_m - (\frac{5}{4}n_b + n'_b) \geq \frac{k_m}{2}$. Observe that, if $j = l$, i.e.,the source and target vertex are both in the $j$'th 2-FCN, hence $d$ can be routed through any path $(b_i^j, m_h^j, b_g^j)$, with $h \in H$, which has congestion less than $5 \cdot OPT - \gamma_d$. This is a contradiction, since we assumed such path does not exists. Hence, $j \neq l$. In this case, let $n_t$ ($n'_t$) be the number of edges incident to any vertex $t_x^h$, with $h \in H$ and $x \in [k_t]$ and congestion at least $5 \cdot OPT$ ($5 \cdot OPT - \gamma_d$ and $5 \cdot OPT$). Observe that each path $(m_h^j, t_x^h, m_h^l)$ must have congestion at least $5 \cdot OPT - \gamma_d$, otherwise $d$ can be routed through $(b_i^j, m_h^j, t_x^h, m_h^l, b_g^l)$, which is a contradiction. Hence, we have $n_t + n'_t \geq |H|k_t \geq (k_m - n_b - n'_b)k_t$. Moreover,

$$\sum_{i=1,\ldots,k_b} \mathcal{F}_{b_i^j} + \sum_{i=1,\ldots,k_b} \mathcal{F}_{b_i^l} \geq n_t(5 \cdot OPT) + n'_t(5 \cdot OPT - \gamma_d) \geq$$

$$\geq 5n_tOPT + 4n'_tOPT = OPT(5n_t + 4n'_t),$$

where $\sum_{i=1,\ldots,k_b} \mathcal{F}_{b_i^j}$ ($\sum_{i=1,\ldots,k_b} \mathcal{F}_{b_i^l}$) is the sum of the flows originated from or directed to a vertex in the $j$'th ($l$'th) FCN of $F$. Let $\mathcal{F}_H = max\{\sum_{i=1,\ldots,k_b} \mathcal{F}_{b_i^j}, \sum_{i=1,\ldots,k_b} \mathcal{F}_{b_i^l}\}$. We have that $2\mathcal{F}_H \geq OPT(5n_t + 4n'_t)$.

Since $\mathcal{F}_H$ must necessarily be split among $k_tk_m$ edges, we have $OPT \geq \frac{\mathcal{F}_H}{k_tk_m}$. Combining this with $2\mathcal{F}_H \geq OPT(5n_t + 4n'_t)$, we obtain $k_tk_mOPT \geq \frac{OPT(5n_t + 4n'_t)}{2}$. Since $\frac{k_m}{2} \geq \frac{5}{4}n_b + n'_b$ and $n_t + n'_t \geq (k_m - n_b - n'_b)k_t$, we have that $2k_tk_m \geq 5n_t + 4n'_t = 4(n_t + n'_t) + n_t \geq 4k_t(k_m - n_b - n'_b) + n_t = 4k_t\left(k_m - \frac{5}{4}n_b - n'_b + \frac{1}{4}n_b\right) + n_t \geq 4k_t\left(\frac{k_m}{2} + \frac{n_b}{4}\right) + n_t \Rightarrow$, which implies that $2k_m \geq 2\left(k_m + \frac{n_b}{4}\right) + \frac{n_t}{k_t} \Rightarrow 0 \geq \frac{n_b}{2} + \frac{n_t}{k_t}$. This is a contradiction since at least $n_b$ or $n_t$ is bigger than 0. In fact, at least one edge have congestion at least $5 \cdot OPT$. This concludes the proof of the lemma. ∎

*Theorem 5.2. After $|\bar{D}|$ reroute operations,* EQUILIBRIUM-ALGO *approximates* MCUF *in 3-FCNs within a factor of* 5.

*Proof:* Let $\bar{D}' = \{d \in \bar{D} | c(p_d) \geq 5 \cdot OPT\}$. By Lemma 5.3, each flow $d \in \bar{D}'$ can be routed through a path $p'$ such that $c(p') \leq 5 \cdot OPT - \gamma_d$ by a single rerouting operation. Once a flow is rerouted, it does no longer belong to $\bar{D}'$. Hence, since $|\bar{D}'| \leq |\bar{D}|$, after at most $\bar{D}$ rerouting operations, each flow $d \in \bar{D}$ is such that $c(p_d) < 5 \cdot OPT$. ∎

## VI. RELATED WORK

Configuring OSPF link weights and ECMP routing have been the subject of extensive research in the past two decades (in a broad variety of contexts: ISP networks, datacenters, and more). Generally speaking, research along these lines has thus far primarily focused on experimental and empirical analyses. We now discuss relevant past studies and their connections to our work. We refer the reader to [21], [22] and [23] for more complete surveys.

**TE with EMCP.** We study TE with ECMP routing within the ("splittable flow") model of Fortz and Thorup [10]. Past work on optimizing ECMP routing mostly examined heuristic approaches (e.g., local search [10], branch-and-cut for mixed-integer linear programming [24], memetic [25] and genetic [26] algorithms) with no provable performance guarantees. [10] proves that MIN-ECMP-CONGESTION is NP-hard and cannot be approximated within a factor of $\frac{3}{2}$. These results leave hope that an (efficient) algorithm for configuring link weights with good (provable) guarantees is possible. Our inapproximability results for MIN-ECMP-CONGESTION, MIN-SUM-COST, and MAX-ECMP-FLOW, shatter this hope (and, in a sense, establish the necessity of heuristics).

**TE with ECMP in datacenters.** The emergence of datacenter networks spurred a renewed interest in interconnection networks [27]. Topologies such as Clos networks [18] and generalized hypercubes [28], [12], [13] have been proposed as datacenter topologies. We compare Clos and hypercube networks from an ECMP routing perspective. Our analysis of Clos networks (Theorem 5.1) supports and explains (i) the experimental results in [14] regarding packet-level traffic splitting in Clos networks, and also (ii) the experimental results in [1] regarding the routing of small (mice) flows via ECMP in Clos networks. Our optimality result for Clos networks shows that the optimal link weight configurations with respect to MIN-ECMP-CONGESTION, MIN-SUM-COST, and MAX-ECMP-FLOW, can be computed independently of the actual demand matrix and can therefore be regarded as "oblivious routing". [19] presents results for oblivious routing in fat tree topologies. Our optimality result for Clos networks can be regarded as a generalization of the result in [19] for oblivious multipath routing in fat trees to more general (Clos) networks and edge capacities, and to other performance metrics (namely, MIN-SUM-COST and MAX-ECMP-FLOW).

**Routing elephant flows in datacenters.** Under ECMP routing, all packets belonging to the same IP flow are routed along the same path. Consequently, a router might map large (elephant) flows to the same outgoing port, possibly leading to load imbalances and throughput losses. Optimizing routes for "unsplittable flows" is shown to be $O(\log n)$-approximable in [29] for general networks. Recent work studies the routing of unsplittable flows in Clos datacenter networks [1], [15],

[6] and experimentally analyzes greedy and other heuristic approaches, e.g., simulated annealing. We initiate the formal analysis of the routing of unsplittable flows in datacenter networks and present upper and lower bounds on the approximability of this task in Clos networks. We present, among other results, a simple, greedy 5-approximation algorithm. We point out that the key idea behind our algorithm (rerouting flows to least loaded paths until reaching an equilibrium) resembles the simulated annealing procedure in Hedera [1] and can be regarded as a first step towards analyzing the provable guarantees of this natural heuristic.

## VII. Conclusion and Future Research

We studied TE with ECMP from an algorithmic perspective. We proved that, in general, not only is optimizing link-weight configuration for ECMP an intractable task, but even achieving a good approximation to the optimum is infeasible. We showed, in contrast, that in some environments ECMP(-like) routing performs remarkably well (e.g., Random Packet Spraying in multi-rooted trees [14], specific traffic patterns). We then turned our attention to the question of optimizing the routing of elephant flows and proved upper and lower bounds on the the approximability of this task. Our results motivate further research along the following lines:

- **ECMP in datacenters.** We showed that TE with ECMP is NP-hard for hypercubes. What about approximating the optimum? Can a good approximation be computed in a computationally-efficient manner? Another interesting question is adapting this result to show similar hardness results for specific hypercube-inspired topologies (e.g., Bcube [12], and MDCube [13]). What about other proposed datacenter topologies, e.g., random graphs ala Jellyfish [30]?
- **Routing elephants.** We presented positive and negative approximability results for routing elephants in folded Clos networks. What is the best achievable approximation-ratio? What are the provable guarantees of simulated annealing (see Hedera [1]) in this context? We believe that research along these lines can provide useful insights into the design of elephant-routing mechanisms.
- **ECMP with bounded splitting.** Consider a model of TE with ECMP in which, to reflect the limitations of today's routers' static hash functions used for ECMP, a router can only split traffic to a destination between a bounded number of links. What can be said about the provable guarantees of TE with ECMP in this model?

## References

[1] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. NSDI*, 2010.

[2] C. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm," RFC 2992, IETF, 2000, http://www.ietf.org/rfc/rfc2992.txt.

[3] J. Moy, "OSPF Version 2," RFC 2328, IETF, 1998, http://www.ietf.org/rfc/rfc2328.txt.

[4] Z. Cao, Z. Wang, and E. W. Zegura, "Performance of Hashing-Based Schemes for Internet Load Balancing," in *Proc. INFOCOM*, 2000.

[5] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Communications Magazine*, vol. 40, pp. 118–124, 2002.

[6] A. G. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: a scalable and flexible data center network," *Commun. ACM*, vol. 54, no. 3, pp. 95–104, 2011.

[7] Cisco, "OSPF Design Guide," 2011, http://www.cisco.com/image/gif/paws/7039/1.pdf.

[8] B. Fortz and M. Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights," in *Proc. INFOCOM*, 2000.

[9] ——, "Optimizing OSPF/IS-IS Weights in a Changing World," *IEEE J.Sel. A. Commun.*, vol. 20, no. 4, pp. 756–767, Sep. 2006.

[10] ——, "Increasing Internet Capacity Using Local Search," *Comp. Opt. and Appl.*, vol. 29, no. 1, pp. 13–48, 2004.

[11] J. R. Lee and A. Naor, "Embedding the Diamond Graph in LP and Dimension Reduction in L1," *Geometric & Functional Analysis*, 2004.

[12] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers," in *Proc. SIGCOMM*, 2009.

[13] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang, "MDCube: A High Performance Network Structure for Modular Data Center Interconnection," in *Proc. CoNEXT*, 2009.

[14] A. A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella, "On the impact of packet spraying in data center networks." in *Proc. INFOCOM*, 2013.

[15] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling Flow Management for High-performance Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 254–265, Aug. 2011.

[16] `http://www.dia.uniroma3.it/~compunet/www/docs/chiesa/ecmp.pdf`.

[17] A. Sridharan, R. Guérin, and C. Diot, "Achieving Near-optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 234–247, Apr. 2005.

[18] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Aug. 2008.

[19] X. Yuan, W. Nienaber, Z. Duan, and R. Melhem, "Oblivious Routing for Fat-tree Based System Area Networks with Uncertain Traffic Demands," in *Proc. SIGMETRICS*, 2007.

[20] I. Holyer, "The NP-Completeness of Edge-Coloring," *SIAM J. Comput.*, vol. 10, no. 4, pp. 718–720, 1981.

[21] A. Altin, B. Fortz, and H. Umit, "Oblivious OSPF Routing with Weight Optimization under Polyhedral Demand Uncertainty," *Netw.*, vol. 60, no. 2, pp. 132–139, Sep. 2012.

[22] J. Rexford, "Route optimization in IP networks," in *Handbook of Optimization in Telecommunications, Springer Science + Business*. Kluwer Academic Publishers, 2006.

[23] P. Siripongwutikorn, S. Banerjee, and D. Tipper, "A Survey of Adaptive Bandwidth Control Algorithms," *IEEE Communications Surveys and Tutorials*, vol. 5, no. 1, pp. 14–26, 2003.

[24] A. Parmar, S. Ahmedy, and J. Sokol, "An Integer Programming Approach to the OSPF Weight Setting Problem," Georgia Institute of Technology, Tech. Rep., 2006.

[25] L. S. Buriol, M. G. C. Resende, C. C. Ribeiro, and M. Thorup, "A Memetic Algorithm for OSPF Routing," 2002.

[26] M. Ericsson, M. G. C. Resende, and P. M. Pardalos, "A Genetic Algorithm for the Weight Setting Problem in OSPF Routing," *Journal of Combinatorial Optimization*, vol. 6, pp. 299–333, 2002.

[27] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[28] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "HyperX: Topology, Routing, and Packaging of Efficient Large-scale Networks," in *Proc. SC*, 2009.

[29] A. Chakrabarti, C. Chekuri, A. Gupta, and A. Kumar, "Approximation Algorithms for the Unsplittable Flow Problem," in *Proc. APPROX*, 2002.

[30] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," in *Proc. NSDI*, 2012.