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

Marco Chiesa, Guy Kindler, Michael Schapira

***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.**

***Index Terms*—Traffic Engineering, Multicommodity Flow, Approximation Algorithms.**

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).

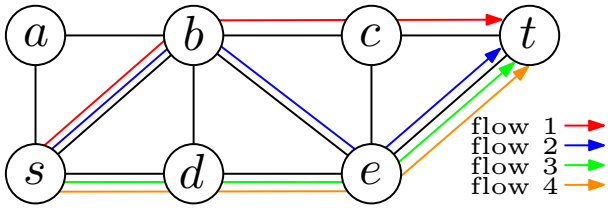

Fig. 1: An illustration of Equal-Cost-MultiPath routing: 4 TCP connections, called "flows 1-4", originate at (source) router s and are destined for (target) router t. All link weights are 1. Observe that (s, b, c, t), (s, b, e, t) and (s, d, e, t) are (all) the induced shortest-paths from s to t. Each 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.

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

M. Chiesa is with Université catholique de Louvain, 1348 Louvain-la-neuve, Belgium. G. Kindler and M. Schapira are with the Hebrew University of Jerusalem, 9190401 Jerusalem, Israel. Email: chiesa@dia.uniroma3.it.

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 we 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 VI. Our results for scheduling elephant flows in folded Clos networks appear in Section VII. We perform an evaluation of our scheduling algorithms for elephant flows in Section VIII. We conclude and present directions for future research in Section X. 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 VII.

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 $f : V \times V \times E \rightarrow \mathbb{R}^+ \setminus \{0\}$. $f(s,t,e)$ 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 define the *congestion* of an edge e as $\frac{f_e}{c_e}$. 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 ϕ 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 α-*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 α-*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 a proof of Theorem 3.1 for the MIN-ECMP-CONGESTION, MAX-ECMP-FLOW, and MIN-SUM-COST objectives.

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 (Section III-A); (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 (Section III-B); and (3) showing that our inapproximability result for MAX-ECMP-DAG implies similar results for both MIN-ECMP-CONGESTION and MAX-ECMP-FLOW (Section III-C).

A. MAX-ECMP-DAG

MAX-ECMP-DAG. We present the following graph-theoretic problem called "MAX-ECMP-DAG" (MED). 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 introduce in Section III-B a new operator that will be used to amplify this inapproximability ratio.

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

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

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).

To prove it, we first show that the MIN-ECMP-CONGESTION problem remains hard even if we have a single source-target pair case (i.e., one single nonzero entry in the demand matrix D). In this case, we observe that an optimal solution for a MIN-ECMP-CONGESTION instance is also an optimal solution for a MAX-ECMP-FLOW instance and vice versa. We then show that every solution for a MAX-ECMP-DAG instance can be translated into an "equivalent" solution

for a MAX-ECMP-FLOW instance (i.e., a link weight assignment) that uses an undirected copy of the original graph. We start from the following theorem, which has been proved by Fortz and Thorup [10].

Theorem 3.3: Given a MIN-ECMP-CONGESTION instance I, with multiple unit flow demands in D, distinguishing between the following two scenarios is NP-Hard:

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

even when each source vertex sends the same amount of traffic to a unique target vertex and each target vertex receives the same amount of traffic from a unique source vertex.

Based on Theorem 3.3, we first prove that the same result holds even in the more restrictive case where there only exists a single source-target pair.

Lemma 3.4: Given a MIN-ECMP-CONGESTION instance I with a single source-target unit flow demand, distinguishing between the following two scenarios is NP-Hard:

- $OPT_{MEC}(I) = 1$.
- $OPT_{MEC}(I) = \frac{3}{2}$.

Proof: Let $I = (G, D)$ be a MEC instance as defined in Theorem 3.3. Let $s_1, \ldots, s_k$ ($t_1, \ldots, t_k$) be the set of source (target) vertices and f be the amount of flow that is sent from a source vertex to a target vertex. For sake of simplicity, we assume that k is a power of 2. Create a copy G' of G and D' of D. Add a new source vertex s into G' and connect it to all vertices $s_1, \ldots, s_k$ with a binary tree rooted at s. Add a new target vertex t and connect it with an edge to all vertices $t_1, \ldots, t_k$. Let $D_{s,t} = 1$, $D_{x,y} = 0$ for $x \neq s$ and $y \neq t$, and set the capacity of each edge of the binary tree incident to a source (target) vertex s_i (t_i) to $\frac{1}{f \cdot k}$ and all the remaining edges of both binary trees to infinite. We then divide each edge capacity by a factor of $f \cdot k$. We now show that $OPT_{MEC}((G, D)) = OPT_{MEC}((G', D'))$. It is easy to see that $OPT_{MEC}((G, D)) \geq OPT_{MEC}((G', D'))$. In fact, if (i) the unit flow demand $D_{s,t}$ is split among every edge in the binary tree that join s to all vertices $s_1, \ldots, s_k$, (ii) each flow from s_i is routed as in the optimal solution for I, and (iii) each flow is routed from each t_i directly to t, then the value of this solution will be equal to $OPT_{MEC}((G, D))$. By observing that an unequal splitting through the binary tree from s to vertices $s_1, \ldots, s_k$ causes a congestion of 2 on the edge that connects a leaf of the binary tree to the corresponding source vertex, our lemma easily derives by Theorem 3.3. ∎

We denote by $I = (G, s, t)$ an instance of both MAX-ECMP-FLOW and MIN-ECMP-CONGESTION with a single source-target unit flow demand from a vertex s to a vertex t. We restrict our attention to those instances that have only optimal solutions with at least an edge with congestion at least 1, i.e., $OPT_{MEC}(I) \geq 1$.

Lemma 3.5: A link weight assignment for a graph G is optimal for an instance (G, s, t) of MIN-ECMP-CONGESTION if and only if it is optimal for an instance (G, s, t) of MAX-ECMP-FLOW.

Proof: It is easy to see that, given an optimal solution for an instance $I = (G, s, t)$ of MIN-ECMP-CONGESTION, by scaling the amount of flow that is sent from s to t by a factor of $\frac{1}{OPT_{MEC}(I)}$, each edge will have congestion at most 1 and the amount of flow sent from s to t is not greater than 1, since $OPT_{MEC} \geq 1$. Hence, $OPT_{MEF}(I) \geq \frac{1}{OPT_{MEC}(I)}$. Vice versa, given an optimal solution for an instance $I = (G, s, t)$ of MAX-ECMP-FLOW, by scaling the amount of flow sent from s to t by a factor of $\frac{1}{OPT_{MEF(I)}}$, each edge will have congestion at most $\frac{1}{OPT_{MEF(I)}}$, since at least an edge in the optimal solution has congestion 1, and a unit of flow will be routed from s to t. Hence, $OPT_{MEC}(I) \leq \frac{1}{OPT_{MEF}(I)}$. ∎

Corollary 3.6: Given a MAX-ECMP-FLOW instance I with a single source-target pair, distinguishing between the following two scenarios is NP-Hard:

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

Proof: It easily follows by Lemma 3.5 and Theorem 3.4 by observing that $OPT_{MEF}(I) = \frac{1}{OPT_{MEC}(I)}$. ∎

We now show that every solution for a MAX-ECMP-DAG instance can be translated in a link weight assignment that is associated with the same flow assignment on the graph. Hence, finding an optimal (s, t)-DAG is equivalent to computing an optimal link weight assignment, which allows us to present our amplification operator in the next section in a simpler way. We now introduce some useful notations. We say that a flow assignment f on a graph G is *realized* by a weight assignment w if by setting the link weights of G as in w, flows are routed according to f. Given a link weight assignment w, let $\mathcal{B}(w)$ denote the oriented subgraph of G that contains the edges that are traversed by a flow. These edges are oriented according to the direction of the traversing flow. Since flows are routed according to the shortest-path criterium, $\mathcal{B}(w)$ is a directed acyclic graph (DAG) with a single source s and a single sink t.

Lemma 3.7: For any arbitrary sub-DAG A with a single source s and a single sink t of a graph G, there exists a link weight assignment w such that $\mathcal{B}(w)$ is equal to A.

Proof sketch: To compute a link weight assignment w that induces A as its shortest-path DAG, it is sufficient to set link weights as follows. Go over the vertices according to the topological order induced by A, $(v_1, \ldots, v_n)$, where $v_1 = t$ and $v_n = s$ and, for each vertex v_i, set the weights of each outgoing link l in A of vertex v_i so that l be on some shortest path from v_i to t. After going over all vertices, set the weights of all links not in A to be sufficiently high so as not to be on *any* shortest path. Observe that this link weight assignment indeed induces A as the resulting shortest-path DAG. ∎

This theorem implies that for any instance $I = (G, s, t)$ of MAX-ECMP-FLOW with $OPT(I) \leq 1$, we have that $OPT_{MF}((G, s, t)) = OPT((G, s, t))$. Hence, combining this with Lemma 3.7, we obtain the following constant inapproximability result for MAX-ECMP-DAG.

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

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

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

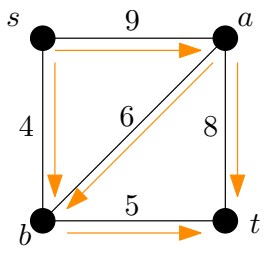

Fig. 2: Graph G_0.

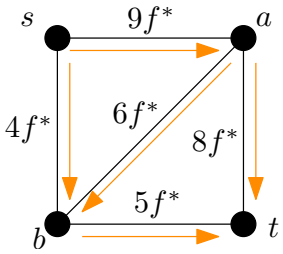

Fig. 3: Abstraction of G_1.

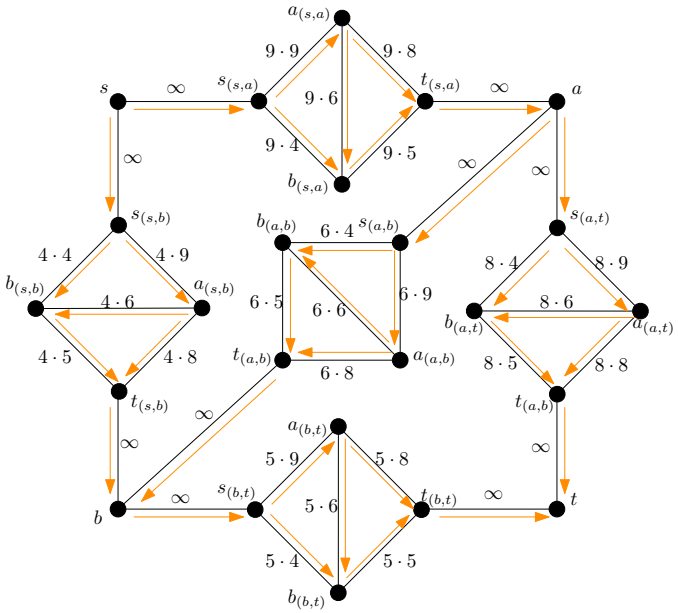

Fig. 4: Graph G_1.

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 ∞.

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 9×6, 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 ∞.

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

Proof: Let $I = \otimes^0 I$ be a MAX-ECMP-DAG instance. Recall that I is a DAG with a single source s and sink t. We prove this lemma by induction on k. Let $\bar{H}_0$ be an optimal solution for $\otimes^0 I$, that is, a sub-DAG of I.

In the base case $k = 0$, we have that $OPT(\otimes^0 I) = OPT(I)^1$, which is true since $\otimes^0 I = I$.

In the inductive case $k > 0$, let $I_k = \otimes^k I$ and $I_{k+1} = \otimes^{k+1} I$. Let $\bar{H}_k$ be an optimal solution for I_k. We prove that there exists a sub-DAG $\bar{H}_{k+1}$ of I_{k+1} such that $OPT(I_{k+1}) = OPT(I)^{k+2}$. First, we prove that $OPT(I_{k+1}) \geq OPT(I)^{k+2}$. Recall that, each edge e of I_k with capacity $c_{I_k}(e) \neq \infty$ is replaced in I_{k+1} by a DAG H_e, where the capacity of each edge of H_e is multiplied by a factor $c_{I_k}(e)$. Consider a solution $\bar{H}_{k+1}$ of I_{k+1} constructed as follows. For each vertex of I_{k+1} that is not contained in any graph H_e (i.e, each vertex in common with I_k), we split the traffic according to the optimal solution in I_k, i.e., for each edge $(x, y) \in E(\bar{H}_k)$ add $(x, s_{(x,y)})$ and $(t_{(x,y)}, y)$ into $\bar{H}_{k+1}$. Moreover,for each subgraph H_e, with $e \in E(I_k)$, we split traffic as $\bar{H}_0$ does in I. Namely, for each subgraph H_e, where $e \in E(\bar{H}_k)$, for each edge $(x, y) \in E(\bar{H}_0)$ add (w_e, y_e) into $E(\bar{H}_{i+1})$. Observe that we can route through H_e a flow that is $OPT(I_0)$ times larger than $c_{I_k}(e)$. Therefore, the maximum flow in I_{k+1} is $OPT(I_0) \cdot OPT(I_k) = OPT(I_0 =) \cdot OPT(I_0)^{k+1} = OPT(I_0)^{k+2}$, which implies $OPT(I_{k+1}) \geq OPT(I)^{k+2}$.

Now, we prove that $OPT(I_{k+1}) \leq OPT(I)^{k+2}$. Suppose, by contradiction, that there exists a sub-DAG $\bar{H}_{k+1}$ of I_{k+1} such that $f_{\bar{H}_{k+1}} > OPT(I)^{k+2}$. Construct a sub-DAG $\bar{H}_k$ of I_k as follows. For each directed edge $(v, u_{(x,y)}) \in E(\bar{H}_{k+1})$, where v is a vertex of I_{k+1} in common with I_k and $u_{(x,y)}$ is the source or target vertex of any subgraph $H_{(x,y)}$, add (v, y) into $E(\bar{H}_k)$ if $y \neq v$, otherwise add (v, x). Since each edge e of I_k can route a flow $OPT(I)$ times smaller than its corresponding subgraph H_e of I_{k+1}, we have that the maximum flow through $\bar{H}_k$ is at least $\frac{f_{\bar{H}_{k+1}}}{OPT(I)} > \frac{OPT(I)^{k+2}}{OPT(I)} = OPT(I)^{k+1}$, which is a contradiction, since, by induction hypothesis, we have that $OPT(I_k) = OPT(I)^{k+1}$. ∎

Lemma 3.8 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 3.8 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

Given an instance H_0 of MAX-ECMP-DAG, let G_k be a copy of $\otimes^k H_0$ with undirected edges. Let s and t be the source and sink vertices of H_0. We denote by I_k an instance (G_k, s, t) of MAX-ECMP-FLOW. We introduce a property of a graph instance that will be used to exploit the amplification technique in MIN-ECMP-CONGESTION and MAX-ECMP-FLOW.

Reversibility. We say that an MEF instance $I = (G, s, t)$ is *non-reversible* if $OPT_{MEF}(I) \geq OPT_{MEF}((G, t, s))$. Similarly, an MEC instance I is non-reversible if $OPT_{MEC}(I) \leq OPT_{MEC}((G, t, s))$.

Lemma 3.9: Let $I = (G, s, t)$ be an MEF instance with $OPT_{MEF} = \{k, \frac{2}{3}k\}$, with $k > 0$. It is possible to construct in polynomial time a non-reversible instance I' such that $OPT_{MEF}(I) = OPT_{MEF}(I')$.

Proof: If G is non-reversible, we create $I' = (G', s', t)$ as a copy of I where s' is a new vertex added into $V(G')$. Moreover, we add four vertices $v_1, v_2.v_3$, and v_4 and connect them to s through a path (v_1, v_2, v_3, v_4, s), with capacity k. Then, we connect s' to each vertex v_1, v_2, v_3, v_4 with an edge of capacity $\frac{1}{4}k$. Observe that, by construction, the maximum flow from s to s' is $\frac{1}{4}k + \frac{1}{8}k + \frac{1}{16}k + \frac{1}{32}k$, while the maximum flow from s' to s is k since s' can send a flow of value $\frac{1}{4}k$ to v_1, v_2, v_3, and v_4, respectively. Hence,

$$\begin{aligned} &OPT_{MEF}(G, s', t) = \\ &= \min\{OPT_{MEF}(G', s', s), OPT_{MEF}(G', s, t)\} = \\ &= \min\{k, OPT_{MEF}(G, s, t)\} = OPT_{MEF}(G, s, t) \geq \\ &\geq \frac{2}{3}k \geq \frac{1}{4}k + \frac{1}{8}k + \frac{1}{16}k + \frac{1}{32}k = \\ &= OPT_{MEF}(G', s, s') \geq OPT_{MEF}(G', t, s'), \end{aligned}$$

which means that I' is non-reversible. ∎

Corollary 3.10: Let $I = (G, s, t)$ be a MEC instance with $OPT_{MEC} = \{k, \frac{3}{2}k\}$, with $k > 0$. It is possible to construct in polynomial time a non-reversible instance I' such that $OPT_{MEC}(I) = OPT_{MEC}(I')$.

Proof: It easily follows by Lemma 3.5. ∎

Relating MAX-ECMP-DAG to MAX-ECMP-FLOW. Given an instance H_0 of MAX-ECMP-DAG such that its undirected copy G_0 is non-reversible. Let s and t be the source and sink vertices of H_0. We say that a flow assignment f in G_0 is *compliant* with H_0 if for every edge $(x, y) \in E(G_0)$, if a flow is routed from x to y, then $(x, y) \in E(H_0)$. We say that a directed acyclic graph H' is an *orientation* of an undirected graph G_0 if at least an optimal flow assignment f of G is compliant with H. We denote $H_k = \otimes^k H_0$, by G_k the undirected copy of H_k, and by I_k an instance (G_k, s, t) of MAX-ECMP-FLOW.

Lemma 3.11: Suppose that in at least one optimal solution for (G_0, s, t) the corresponding flow assignment is compliant with H_0. Then, $OPT(H_k) = OPT_{MEF}(I_k)$.

Proof: We prove it by induction. In the base case $k = 0$, the statement of the lemma holds since instance I_0 is such that it has an optimal solution that is compliant with H_0. In the inductive case $k > 0$, by Lemma 3.7 we know that $OPT_{MEF}(I_k) = OPT(H_k) = OPT(H_0)^{k+1}$ and $OPT_{MEF}(I_{k+1})$ is at least $OPT(H_{k+1}) = OPT(H_0)^{k+2}$. We want to show that $OPT_{MEF}(I_{k+1}) \leq OPT(H_0)^{k+2}$. Suppose, by contradiction, that there exists a sub-DAG $\bar{H}_{k+1}$ of I_{k+1} such that $f_{\bar{H}_{k+1}} > OPT(I)^{k+2}$. Construct a sub-DAG

$\bar{H}_k$ of I_k as follows. For each directed edge $(v, u_{(x,y)}) \in E(\bar{H}_{k+1})$, where v is a vertex of I_{k+1} in common with I_k and $u_{(x,y)}$ is the source or target vertex of any subgraph $H_{(x,y)}$, add (v, y) into $E(\bar{H}_k)$ if $y \neq v$, otherwise add (v, x). Recall that, since I^0 is non-reversible, for each graph H_e, we have $OPT_{MEF}(H_e, t_e, s_e) \leq OPT_{MEF}(H_e, s_e, t_e) = OPT_{MEF}(H_0, s, t) \cdot c_{I_k}(e)$, which means that, for each edge e of H_k, we can route at least a flow $OPT_{MEF}(I_0)$ times smaller than in H_e. Hence, solution $\bar{H}_k$, induces a maximum flow through H_k of at least $\frac{OPT_{MEF}(I_{k+1})}{OPT_{MEF}(I_0)} > \frac{OPT_{MEF}(I_0)^{k+2}}{OPT_{MEF}(I_0)} = OPT_{MEF}(I_0)^{k+1}$ units, which is a contradiction, since, by induction hypothesis, we have that $OPT_{MEF}(I_k) = OPT_{MEF}(I_0)^{k+1}$. ∎

By Lemma 3.5. we have the following corollary.

Corollary 3.12: $OPT(H_k) = \frac{1}{OPT_{MEC}(I_k)}$.

We present the following lemma, which concludes the proof.

Lemma 3.13: For any $\alpha > 1$, if MAX-ECMP-DAG is NP-hard to approximate within a factor of α then

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

Proof: Suppose, by contradiction that there exists an $\alpha > 0$, such that MAX-ECMP-FLOW can be approximated within a factor of α, i.e., there exists a polynomial time algorithm $\mathcal{A}$ that, given an instance I of MAX-ECMP-FLOW, returns a solution of value $\mathcal{A}(I) \geq \frac{OPT_{MF}(I)}{\alpha}$. We can construct a α-approximation algorithm for MAX-ECMP-DAG as follows. Let $I_0 = (H_0, s, t)$ be a MAX-ECMP-DAG instance used in Lemma 3.6 such that its undirected copy G_0 is non-reversible. By Lemma 3.9, we know that such instance must exists. Moreove, we have that $OPT(I_0)$ is either 1 or $\frac{2}{3}$. Further, it was proved in [8] that it is easy to compute an orientation of G_0 such that at least one optimal solution is compliant with it. Now, let c be an integer such that $\left(\frac{2}{3}\right)^c < \alpha$. Let $H_c = \otimes^c H_0$, denote by G_c the undirected copy of H_c, and by I_c an instance (G_c, s, t) of MAX-ECMP-FLOW. By Lemma 3.11, we have that $OPT(H_c) = OPT_{MF}(I_c)$. Now, if $OPT(H_0) = 1$, we have that $\mathcal{A}((G_c, s, t)) \geq \alpha$. Otherwise, if $OPT(H_0) = \frac{2}{3}$, since $OPT_{MF}(I_c) = OPT(H_c) = \left(\frac{2}{3}\right)^c$, we have that $\mathcal{A}(I_k) \leq \left(\frac{2}{3}\right)^c < \alpha$. Hence, $\mathcal{A}$ can be used to distinguish, in polynomial time, between MAX-ECMP-DAG instances with optimal value 1 or $\frac{2}{3}$, which is a contradiction to Theorem 3.2.

By Lemma 3.5, the same result also holds for MIN-ECMP-CONGESTION. ∎

IV. SUM OF LINK COSTS INAPPROXIMABILITY

We now turn our attention to the well-studied MIN-SUM-COST problem, presented in [8], where each link has a cost that depends on the amount of flow routed through it, and the goal is to minimize the sum of the costs acoss links. As in [5], [8], [9], we consider the individual-link-cost function $\phi(x) = 2^x - 1$. The objective is to route flow so as to minimize the expression

$$\min \sum_{e \in E(G)} \phi\left(\frac{f_e}{c_e}\right)$$

.

We show that approximating the optimum within any constant factor is NP-Hard. To this end, we again exploit our amplification technique, which uses the operator $\otimes$.

We first introduce and study a related problem, called MIN-CONGESTED-EDGES. The goal is to minimize the number of edges have congestion, that is, value $\frac{f_e}{c_e}$, at least $\frac{3}{2}$. We then show how to leverage our $\otimes$ amplification technique to amplify the gap between two different classes of instances of MIN-CONGESTED-EDGES, as we did for the MAX-ECMP-DAG problem in the previous section. Finally, we relate MIN-CONGESTED-EDGES to MIN-SUM-COST, concluding that the the latter is not approximable within any constant factor (Theorem 4.4).

MIN-CONGESTED-EDGES problem. An instance I of this problem is a graph G, a source vertex s, and a target vertex t, exactly as in the MEC and the MEF problems.

In the reduction from 3-SAT used in [10] to prove that MIN-ECMP-CONGESTION is not approximable within a factor of $\frac{3}{2}$, a SAT formula F with n variables is transformed into an instance $I = ((G, s, t), \cdot)$ of MIN-ECMP-CONGESTION such that, if a variables assignment satisfies a clause c, then the edge e_c associated with clause c is such that $\frac{f_{e_c}}{c_{e_c}} \leq 1$, otherwise $\frac{f_{e_c}}{c_{e_c}} = \frac{3}{2}$. Since the reduction is from 3-SAT, we can use the following well-known inapproximability result (Theorem 4.1) to prove that also in a slightly modified Fortz and Thorup construction, at least a certain amount of edges must be congested (Lemma 4.2).

Theorem 4.1: [18]. For any $\epsilon > 0$, MAX-3-SAT is $(\frac{7}{8} + \epsilon)$-hard to approximate.

We omit the proof of the following lemma, which is based on Theorem 4.1 and on a straightforward modification of the Fortz and Thorup construction.

Lemma 4.2: There exists two constants $\alpha > 1$ and $p > 0$ such that, given a congestion threshold $C = \frac{3}{2}$, it is NP-Hard to approximate MIN-CONGESTED-EDGES within a factor of α even if the input instance I is "non-reversible", in its optimal solution either all edges have congestion at most 1 or at least a fraction p of its edges have congestion at least C, and an orientation of I is given in input.

In MIN-CONGESTED-EDGES, an instance $I = (G, s, t)$ is *non-reversible* if, in every optimal solution of (G, t, s) at least $p > 0$ edges have congestion at least $\frac{3}{2}$.

We now prove the following key lemma that, given an instance I, provide a lower bound on the number of edges that are "heavily" congested in $\otimes^k I$, with $k \in \mathbb{N}$.

Let $I = (G, s, t)$ be a non-reversible MIN-SUM-COST instance such that it only admits solutions where at least a fraction $p > 0$ of its edges have congestion at least C. Let H be a directed copy of G such that there exists at least an optimal flow assignment for I that is compliant with H. We denote by G_k the undirected copy of $\otimes^k H$ and by $I_k = (G_k, s, t)$.

Lemma 4.3: For every $k \geq 0$, every solution for (G_k, s, t) is such that at least a fraction p^{k+1} of the edges of G_k have congestion at least C^{k+1}.

Proof: We prove it inductively on k. In the base case $k = 0$, the statement trivially holds. In the inductive step $k > 0$, by inductive hypothesis, in every solution of I^k at least $p^{k+1}|E(G_k)|$ edges have congestion at least C^{k+1}. We want to prove that in every solution of I_{k+1} at least $p^{k+2}|E(G_{k+1})|$ of the edges have congestion at least C^{k+2}. Suppose, by contradiction, that there exists an optimal solution $\bar{A}$ of I_{k+1} (i.e., a sub-DAG of $\otimes^k H$ with a source s and a sink t) such that less than $p^{k+2}|E(G_{k+1})|$ edges have congestion at least C^{k+2}. We now construct an optimal sub-DAG solution A_k of I_k from $\bar{A}$ exactly as we did in the proof of Lemma 3.8, i.e., for each vertex in common between G_k and G_{k+1}, we split traffic in the same way.

Recall that, each edge e of G_k with capacity $c_{G_k}(e) \neq \infty$ is replaced by a graph $G_e = G$ in G^{k+1}, where the capacity of each edge of G_e is multiplied by $c_{G_k}(e)$. Observe that, by definition of G, we have that at least a fraction p of the edges in $E(G)$ have a congestion of C, when one unit of traffic is routed from s to t in G. As a consequence, by definition of G_e, we know that at least a fraction p of its edges have congestion $C_e \geq Cf_e$, where f_e is the amount of flow routed from s_e to t_e in G_{k+1}. By construction of A_k, we have that edge $e \in E(G_k)$ is also traversed by a flow f_e, which means that its congestion is $\frac{C_e}{C}$. By a simple counting argument, there only exist at least $(1 - p^{k+1})|E(G_k)|$ subgraphs G_e such that for each of them less than $p|E(G)|$ edges have congestion at least C^{k+2}. This implies, that the flow assignment associated with A_k is such that at least $(1 - p^{k+1})|E(G_k)|$ edges have congestion at most $\frac{C_e}{C} < \frac{C^{k+2}}{C} = C^{k+1}$, which is a contradiction since, by inductive hypothesis, in any solution of I_k at least $p^{k+1}|E(G_k)|$ edges have congestion at least C^{k+1}. ∎

We now prove that MIN-SUM-COST is inapproximable within any constant factor. We consider the two class of instances of Lemma 4.2. We then leverage our construction technique based on operator $\otimes$ on these instances. As a consequence, by Lemma 3.12 and Lemma 4.3, the gap between the optimal sum of link costs can be set arbitrary large.

Theorem 4.4: It is NP-Hard to approximate the MIN-SUM-COST problem within any constant factor.

Proof: Suppose that there exists an α-approximation algorithm for a certain constant α. Let $I = (G, s, t)$ be a non-reversible instance of MIN-CONGESTED-EDGES used to prove the NP-Hard-ness in Lemma 4.3, i.e., in any optimal solution of I either (i) all edges have congestion at most 1 or (ii) at least a fraction p of the edges have congestion at least $C = \left\{\frac{3}{2}\right\}$. Let H be a directed copy of G such that there exists at least an optimal flow assignment of I that is compliant with H.

We now leverage our result for MIN-CONGESTED-EDGES to get an estimate of the value of an optimal solution of the MIN-SUM-COST problem on an instance constructed based on I using our operator $\otimes$.

In case (i), by Lemma 3.12, each edge of G_k in the optimal solution of I^k has congestion at most 1. Hence, $\sum_{e \in E(G_k)} \phi\left(\frac{f_e}{c_e}\right) \leq \phi(1)|E(G_k)| = |E(G_k)|$. In case (ii), by Lemma 4.3, there exists at least a fraction p^{k+1} of the edges of G_k that have congestion at least C^{k+1}. Hence, $\sum_{e \in E(G_k)} \phi\left(\frac{f_e}{c_e}\right) \geq p^{k+1}|E(G_k)|\phi\left(\left(\frac{3}{2}\right)^{k+1}\right) = p^{k+1}|E(G_k)|2^{\left(\frac{3}{2}\right)^{k+1}-1}$. Hence, the value of an optimal solution in case (ii) is at least $2^{\left(\frac{3}{2}\right)^{k+1}-1}p^{k+1}$ times higher than the value of an optimal solution in case (i). This quantity can be made larger than α, for any $\alpha \geq 1$, by carefully selecting a certain $k > 0$. This implies that, an α-approximation algorithm for MIN-SUM-COST can be exploited to distinguish between the two class of instances, which is a contradiction because of Lemma 4.2. ∎

V. NON-CONSTANT (ALMOST POLYNOMIAL) INAPPROXIMABILITY FACTORS

Theorem 3.1 shows that both MIN-ECMP-CONGESTION and MAX-ECMP-FLOW cannot be approximated with any constant factor unless P=NP. However, if one is willing to use a slightly stronger assumption than $P \neq NP$, namely that NP is not contained in 'quasi-polynomial' time, then a stronger hardness result is attainable (see below). Again, this result is achieved via the repeated use of our gap-amplification technique (see, e.g., [19] for a similar approach).

To make our statement formal, we define the quasi-polynomial time family to be the set of decision problems that have an $n^{(\log n)^\beta}$-time solution, where n denotes the size of the instance and β is any positive constant.

Theorem 5.1: For any $\epsilon > 0$, MAX-ECMP-FLOW is hard to approximate within factor $\left(\frac{3}{2}\right)^{(\log n)^{1-\epsilon}}$, where n is the number of edges of the input graph, unless NP is in quasi polynomial time.

Note that if one assigns the value 0 to the term ϵ in the expression for the hardness-of-approximation factor above, then it becomes a constant power of n. But since the theorem requires that $\epsilon > 0$, one can interpret the hardness factor as being "almost-polynomial" in n – a power of n that slowly decreases to 0 as n grows. Before we prove Theorem 5.1 let us start with the following technical lemma.

Lemma 5.2: Let I be a MAX-ECMP-DAG instance. Then $|E(\otimes^k I)| \leq |E(I)|^{k+2}$.

Proof: Let $|E(I)|$ be the number of edges of I. The number of edges of $\otimes^k I$ is $|E(\otimes^k I)| = |E(I)|^{k+1} + 2(|E(I)|^k + \ldots + |E(I)|) \leq |E(I)|^{k+1} + 2|E(I)|^{k+1} \leq |E(I)|^{k+2}$, where in the last inequality we assumed that $|E(I)| \geq 2$. ∎

We are now ready to prove Theorem 5.1.

Proof of Theorem 5.1: We repeat the construction as in Lemma 3.8, except that we increase the value of k. Consider a given MAX-ECMP-DAG instance $I_0 = (G, s, t)$, whose optimal solution is either 1 or $\frac{3}{2}$. We now pick $k = \lceil(\log|E(G)|)^\gamma\rceil$, for some constant $\gamma > \frac{1-\epsilon}{\epsilon}$. By Lemma 5.2 we have that

$$|E(\otimes^k I)| \leq |E(I_0)|^{k+2}, \tag{1}$$

and thus $\otimes^k I$ can be constructed from I_0 in quasi-polynomial time. By Lemma 3.8, we have that $OPT(\otimes^k I)$ is either 1 or $\left(\frac{2}{3}\right)^{k+1}$, depending on the maximal flow in the original instance I. If we could get a polynomial time approximation

for MAX-ECMP-DAG within factor $\left(\frac{2}{3}\right)^{k+1}$ on $\otimes^k I$ (here we mean polynomial time in the size of $\otimes^k I$), we could determine whether $OPT(I)$ is 1 or $\frac{2}{3}$. Together with the construction of $\otimes^k I$ this would take quasi-polynomial time, and would be a contradiction of Theorem 3.2 if we assume that NP is not contained in quasi-polynomial time.

We thus have that it is hard to get a polynomial time approximation within $\left(\frac{2}{3}\right)^{k+1}$ for an instance the size of $\otimes^k I$. Let us now recompute the value of k as a function of the size of $\otimes^k I$. Using Equation 1, we have

$$|E(I_0)|^{k+2} \geq |E(\otimes^k I)|$$

$$(k+2)\log|E(I_0)| \geq \log|E(\otimes^k I)|.$$

Since $\log|E(I_0)| \leq k^{\frac{1}{\gamma}} \leq (k+2)^{\frac{1}{\gamma}}$, we have that

$$(k+2)(k+2)^{\frac{1}{\gamma}} \geq \log|E(\otimes^k I)|$$

$$(k+2)^{\frac{\gamma+1}{\gamma}} \geq \log|E(\otimes^k I)|$$

$$k \geq (\log|E(\otimes^k I)|)^{\frac{\gamma}{\gamma+1}} - 2$$

$$k \geq (\log|E(\otimes^k I)|)^{1-\epsilon} - 2$$

which implies that MAX-ECMP-FLOW is not approximable within a factor of

$$\left(\frac{3}{2}\right)^{k+1} \geq \left(\frac{3}{2}\right)^{(\log|E(\otimes^k I)|)^{1-\epsilon}},$$

unless NP is in quasi polynomial time, which proves the statement of the theorem. ∎

VI. 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 VII.

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 $1,\ldots,n-1$ of F^n:** Create, for some chosen $k>0$, k duplicates of F^{n-1}: $F_1^{n-1},\ldots,F_k^{n-1}$. Set stage $i=1,\ldots,n-1$ of F^n to be the union of the i'th stages of $F_1^{n-1},\ldots,F_k^{n-1}$. Create an edge between two vertices in stages $1,\ldots,n-1$ 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 $v_{i,1},\ldots,v_{i,r}$ 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 $\bigcup_i\{v_{i,1}\ldots,v_{i,r}\}$. 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 $\{v_{i,1},\ldots,v_{i,r}\}$.

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 [20], [6], [21]). 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 6.1: 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 $1,\ldots,l$ (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 $F_1^{l-1},\ldots,F_m^{l-1}$ 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 $i=1,\ldots,m$ denotes the set of vertices in the last stage of F_i^{l-1}. 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

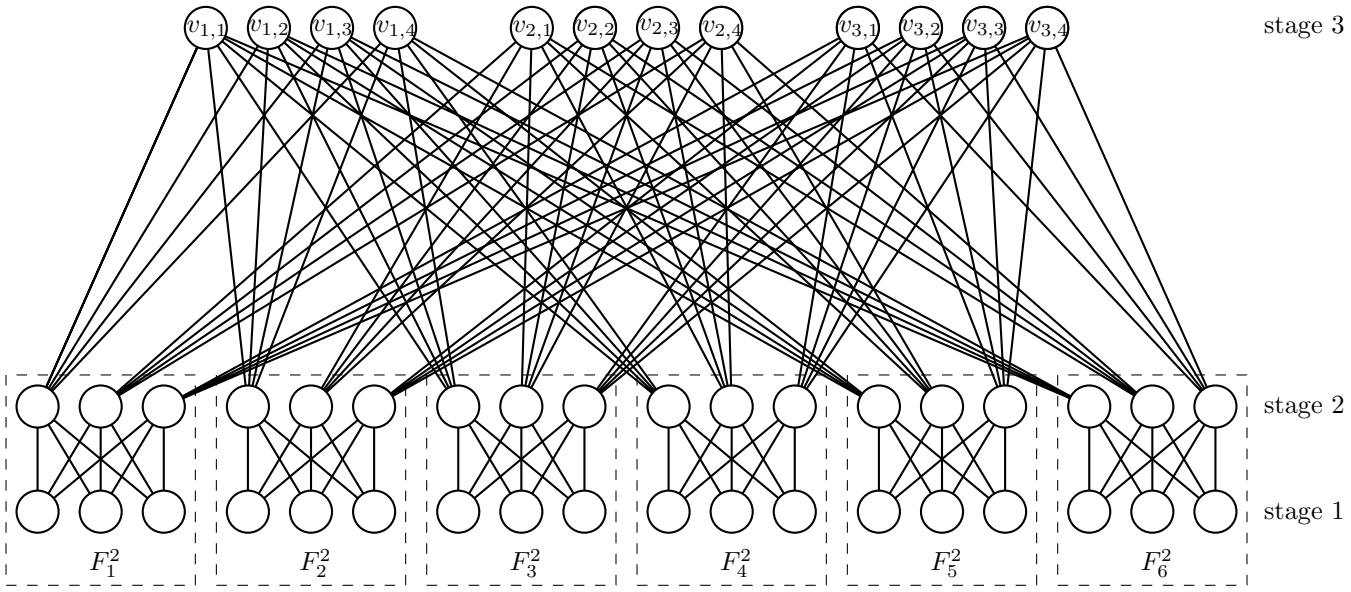

Fig. 5: A 3-FCN constructed by interconnecting six 2-FCNs.

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 source 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 6.2: Computing the optimal flow with respect to MIN-ECMP-CONGESTION in hypercubes is NP-hard.

VII. 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,γ) 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,γ), a *s*ingle 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 [22].

Theorem 7.1: 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. [20] 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 7.2: After $|\bar{D}|$ reroute operations, EQUILIBRIUM-ALGO approximates MCUF in 3-FCNs within a factor of 5.

Theorem 7.1 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 X). 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^j_i (m^j_i) the i'th vertex in the first (second) stage of the j'th FCN. Each vertex m^j_i is connected to vertices $t^j_1,\ldots,t^j_{k_t}$ 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 γ_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. For the sake of simplicity, we assume that all edges has equal capacity. Since the capacity scales both the value of the optimal solution and the value of the solution computed by EQUILIBRIUM-ALGO by the same factor, we can assume that all edges have capacity fo 1.

Lemma 7.3: Let $d \in \bar{D}$ be a flow demand such that $c(p_d) > 5 \cdot OPT$. 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^j_i and b^l_g, 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 $c(p_d) > 5 \cdot OPT \geq 5\gamma_d$. Let n_b (n'_b) be the number of edges incident to b^j_i plus the number of edges incident to b^l_g that have congestion greater than $5 \cdot OPT$ (greater than $5 \cdot OPT - \gamma_d$ 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, $\mathcal{F}_{b^j_i} + \mathcal{F}_{b^l_g} > n_b(5 \cdot OPT) + n'_b(5 \cdot OPT - \gamma_d) \geq$

$5n_bOPT + n'_b(5 \cdot OPT - OPT) = 5n_bOPT + 4n'_bOPT$. Let $\mathcal{F}_* = \max\{\mathcal{F}_{b_i^j}, \mathcal{F}_{b_g^l}\}$. We have $2\mathcal{F}_* > 5n_bOPT + 4n'_bOPT$. 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 $k_mOPT > \frac{5n_bOPT+4n'_bOPT}{2} \Rightarrow k_m > \frac{5n_b+4n'_b}{2} \Rightarrow \frac{k_m}{2} > \frac{5}{4}n_b + n'_b$.

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 $5 \cdot OPT - \gamma_d$ and $\bar{H}$ be the set of indices $\bar{h} \in [k_m]$ not contained in H, i.e., the set of indices $\bar{h} \in [k_m]$ such that $(b_i^j, m_{\bar{h}}^j)$ or $(b_g^l, m_{\bar{h}}^l)$ has congestion greater than $5 \cdot OPT - \gamma_d$. Observe that $|\bar{H}| \leq n_b + n'_b$ and $|H| = k_m - |\bar{H}|$, which implies that $|H| = k_m - |\bar{H}| \geq k_m - (n_b + n'_b) \geq k_m - (\frac{5}{4}n_b + n'_b)$. Further, since $\frac{k_m}{2} > \frac{5}{4}n_b + n'_b$, we have that $k_m - (\frac{5}{4}n_b + n'_b) > \frac{k_m}{2}$, which implies that $|H| > \frac{k_m}{2}$. 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 at most $5 \cdot OPT - \gamma_d$. 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 greater than $5 \cdot OPT$ (greater than $5 \cdot OPT - \gamma_d$ and at most $5 \cdot OPT$). 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,

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

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

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 $2\mathcal{F}_H > OPT(5n_t + 4n'_t)$.

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 $2\mathcal{F}_H > OPT(5n_t + 4n'_t)$, we obtain $k_tk_mOPT > \frac{OPT(5n_t+4n'_t)}{2}$. Since $\frac{k_m}{2} > \frac{5}{4}n_b + n'_b$ and $n_t + n'_t \geq (k_m - n_b - n'_b)k_t$, we have that $2k_tk_m > 5n_t + 4n'_t = 4(n_t + n'_t) + n_t \geq 4k_t(k_m - n_b - n'_b) + n_t = 4k_t\left(k_m - \frac{5}{4}n_b - n'_b + \frac{1}{4}n_b\right) + n_t \geq 4k_t\left(\frac{k_m}{2} + \frac{n_b}{4}\right) + n_t$, which implies that $2k_m > 2\left(k_m + \frac{n_b}{4}\right) + \frac{n_t}{k_t} \Rightarrow 0 > \frac{n_b}{2} + \frac{n_t}{k_t}$— a contradiction. This concludes the proof of the lemma. ■

Theorem 7.2. 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 7.3, 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 $c(p_d) < 5 \cdot OPT$. ■

Corollary 7.4: If all flows have equal size, then EQUILIBRIUM-ALGO is a 4-approximation algorithm and it runs in polynomial time if it is halted after at most $|D|$ reroute operations.

VIII. SIMULATIONS

In this section, we perform an evaluation of EQUILIBRIUM-ALGO by comparing it against the simpler ECMP routing, which is a static routing algorithm, and the more complex HEDERASA [1], a simulated-annealing-based routing algorithm designed to dynamically reroute flows within a data center network. We consider Clos networks with 3 stages and the same number of ports per switch. We recall that the end-hosts servers are connected to the first layer of the Clos network. Our simulation are reproducible and publicly available [23].

We first describe how HEDERASA works. While EQUILIBRIUM-ALGO assigns a core switch (i.e., a top-layer switch) to each large flow, HEDERASA assigns a core switch to each end-host destination, hence routing the whole traffic on a per-destination basis. To minimize the chances of congestion among large flows, HEDERASA assigns a different core switch to each end-host destination within the same 2-FCN subnetwork, commonly referred as a POD of the Clos network. HEDERASA periodically swaps the core switches assigned to the end-host destinations based on an estimate of the traffic demands, which is computed by an external demand estimation component. In our simulations, we let each routing algorithm access the same demand estimation component, thus providing a fare comparison between the routing algorithm components, independent of the specific demand estimator algorithm. Based on the benchmark evaluation of the demand estimation process described in [1], we assume that the size of a flow is revealed to the routing algorithm after 10ms, which is the average time required in [1] to detect an elephant flow in a Clos network with no more than 8192 hosts.

We run the network simulator used to assess the quality of HEDERASA, as described in [1]. This simulator receives as input a traffic pattern consisting of a set of traffic demands of different sizes. We generate traffic according to typical datacenter traffic patterns [6], in which most of the flows are small, yet the few large large flows account for the vast majority of the bits sent throughout the network. We generate flows based on a Poisson arrival process. The simulator models data communication using a flow model, where the size of a flow represents the bandwidth used by that flow at a specific time. TCP slow start and AIMD are modeled by increasing and decreasing the size of a flow depending on whether there is a link that saturated its bandwidth on the path where the flow is routed. For a more detailed description of the simulator and a discussion of its limitations, we refer the reader to [1].

We ran each simulation for the equivalent of 30 seconds and measured the average throughput achieved by ECMP, EQUILIBRIUM-ALGO, and HEDERASA in the interval $[5, 25]$. To make a fair comparison between EQUILIBRIUM-ALGO and HEDERASA we use the same initial per-destination routing for both of them. Namely, we initially route flows as previously described in the HEDERASA approach, where each destination end-host is assigned a different core switch.

Our analysis, shows that both EQUILIBRIUM-ALGO and HEDERASA outperform ECMP by roughly a 10% factor both

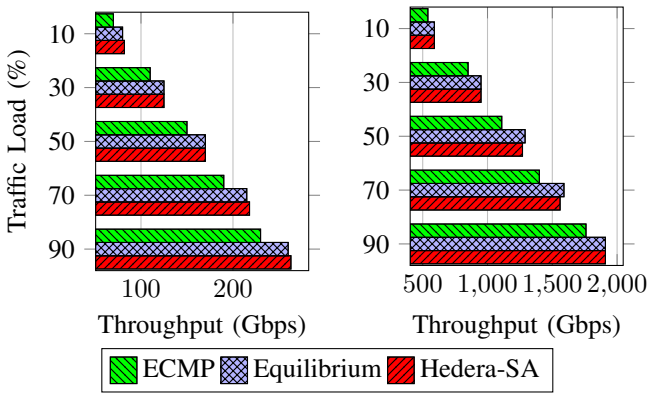

Fig. 6: Average throughput in a 3 stages Clos network with 8 (left) and 16 (right) ports per switch, respectively.

in a Clos network with 128 and 1024 end hosts (see Fig. 6). We observe that the higher complexity of running a simulated annealing procedure provides little or no benefits compared to the simpler EQUILIBRIUM-ALGO routing algorithm. In addition, EQUILIBRIUM-ALGO reroutes less flows than HEDERASA since it only reroutes large flows.

IX. 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 [24], [25] and [26] 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 [27], memetic [28] and genetic [29] 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 [30]. Topologies such as Clos networks [20] and generalized hypercubes [31], [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 6.1) supports and explains (i) the experimental results in [14], [32] 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". [21] 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 [21] 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 [33] for general networks. Recent work studies the routing of unsplittable flows in Clos datacenter networks [1], [15], [6], [34] 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.

X. 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 [35]?
- **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?

ACKNOLEDGEMENTS

This research was partially supported by the Israeli Center for Research Excellence in Algorithms (I-CORE) and the Israel Science Foundation (ISF grant 420/12).

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.*, 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] J. Håstad, "Some optimal inapproximability results," *J. ACM*, vol. 48, no. 4, pp. 798–859, Jul. 2001. [Online]. Available: http://doi.acm.org/10.1145/502090.502098

[19] M. Bellare, O. Goldreich, and M. Sudan, "Free bits, pcps and non-approximability – towards tight results," 1996.

[20] 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.

[21] 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.

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

[23] `github.com/marchiesa/equilibrium-data-center`.

[24] 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.

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

[26] 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.

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

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

[29] 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.

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

[31] 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.

[32] J. Cao, R. Xia, P. Yang, C. Guo, G. Lu, L. Yuan, Y. Zheng, H. Wu, Y. Xiong, and D. A. Maltz, "Per-packet load-balanced, low-latency routing for clos-based data center networks," in *CoNEXT*, 2013.

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

[34] W. Wang, Y. Sun, K. Zheng, M. A. Kâafar, D. Li, and Z. Li, "Freeway: Adaptively isolating the elephant and mice flows on different transmission paths," in *ICNP 2014*.

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