Traffic Engineering with ECMP: 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 first consider the standard "splittable-flow" model of TE with ECMP, put forth in [18]. We settle a long-standing open question by proving that, in general, even *approximating* the optimal link-weight configuration for ECMP within *any* constant ratio is an intractable feat. We also initiate the *analytical* study of TE with ECMP on *specific* network topologies and, in particular, datacenter networks. We prove that while TE with ECMP remains suboptimal and computationally-hard for hypercube networks, ECMP can, in contrast, provably achieve optimal traffic flow for the important category of folded Clos networks. We next investigate the approximability of TE with ECMP in the more realistic "unsplittable-flow" model and present upper and lower bounds for scheduling "elephant" flows on top of ECMP (as in, e.g., Hedera [4]). Our results complement and shed new light on past experimental and empirical studies of the performance of TE with ECMP.**

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) [23] via static link-weight configuration.

Most large IP networks run Interior Gateway Protocols, e.g., Open Shortest Path First (OSPF) [25], to compute all-pairs shortest-paths between routers based on configurable static link weights. 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 [8]. See Figure 1 for an illustration of shortest-path routing and ECMP traffic splitting on a simple network topology. Hence,

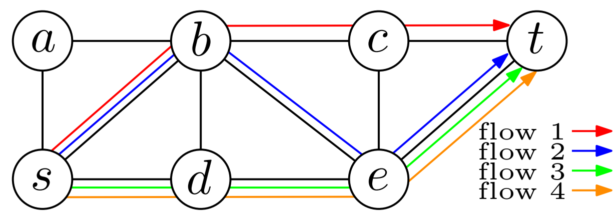

Fig. 1: An illustration of Equal-Cost-MultiPath routing: 4 TCP connections, called "flows 1-4", originate at (source) router s and are destined for (target) router t. All link weights are 1. Observe that (s, b, c, t), (s, b, e, t) and (s, d, e, t) are (all) the induced shortest-paths from s to t. Each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

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).

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 [15] and for datacenter networks [19]), 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 [10]. 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 [16], [17], [18], and the common 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 (with respect to this objective) within *any* constant ratio. We show that this inaproximability result extends to other metrics of interest, e.g., maximizing total throughput and minimizing the sum of (exponentially-increasing) link costs (introduced in [16]). 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 [24]) 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 [19]) and hypercubes (BCube [20], MDCube [31],). 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 constrast, that in hypercubes computing the optimal link weights for ECMP is NP-hard.

Our optimality result for folded Clos networks supports the experimental results in [4] regarding the routing of small (mice) flows via ECMP in Clos networks. Our result also supports the experimental findings in [13]. 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. 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 [13], [4]. [13] 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). [13] shows, via extensive simulations, that "ECMP-like" traffic splitting at packet-level granularity leads to significantly better load-balancing of traffic and, consequently, better network performance in folded Clos networks ("fat-trees"). We point out that as the splittable-flow model in [13] captures fine-grained traffic splitting, our optimality result for folded-Clos networks provides a strong theoretical justification for this claim.

Optimizing the routing of elephant flows. As discussed above, ECMP's flow-level traffic splitting can result in poor utilization of network resources and in undesirable phenomena such as link congestion. Beyond transitioning to ECMP traffic splitting at packet-level, researchers have also examined other possible approaches to alleviating this. Recent studies, e.g., Hedera [4] and DevoFlow [11], call for dynamically scheduling elephant flows in datacenter (folded Clos) networks so as to minimize traffic imbalances (while still routing small, "mice" flows via 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 our inapproximability result for optimizing link-weight configuration in Section III. We present 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 [1].

II. EMCP ROUTING MODEL

We now present the standard model for TE with ECMP from [18]. We refer the reader to [16], [17], [18] 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 $f : V \times V \times E \rightarrow \mathbb{R}^+ \setminus \{0\}$. $f(s,t,e)$ 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 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 $s,t \in V$ $\Sigma_{e \in E_s} f(s,t,e) = \Sigma_{e \in E_t} f(s,t,e) = D_{s,t}$. (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" $w : E \to \mathbb{R}^+ \setminus \{0\}$. 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 optiomization goal that has been studied in the context of TE with ECMP is MIN-SUM-COST [16], [17], [18], [30]: 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 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 [1].

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 [18]. We shall later amplify this inapproxmability ratio.

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

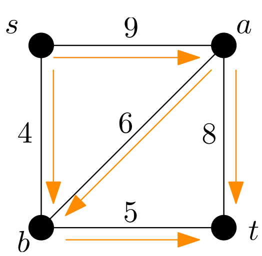

Fig. 2: Graph G_0.

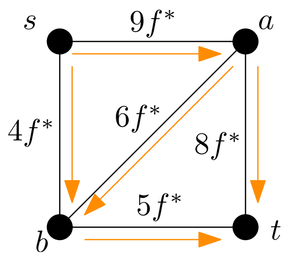

Fig. 3: Abstraction of G_1.

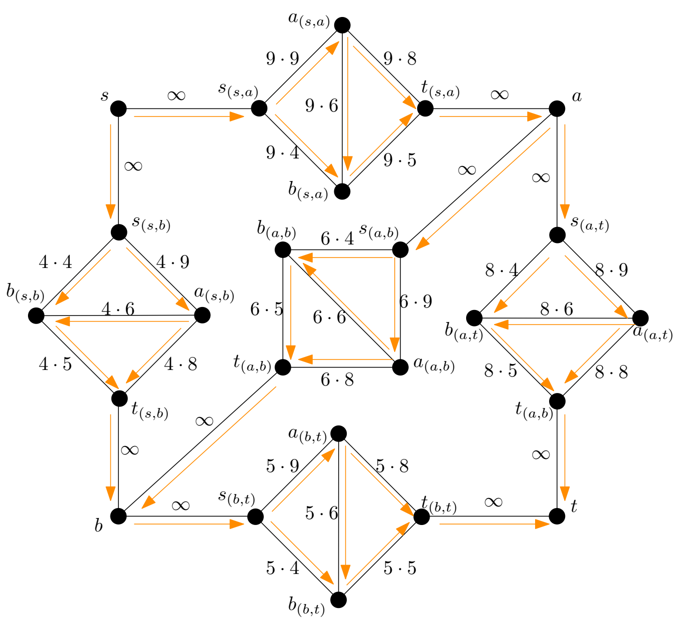

Fig. 4: Graph G_1.

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

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

Proof of Theorem 3.2 can be found in Appendix B. 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: 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 amplifyed 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 3.3: Let I be an instance of MAX-ECMP-DAG. $OPT(\otimes^k I) = (OPT(I))^{k+1}$ for any integer $k > 0$.

Proof of Lemma 3.3 is in Appendix C. 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, MAX-ECMP-FLOW, *and* MIN-SUM-COST

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 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.

Our proof of this lemma also involves the application of the $\otimes$ operator and is deferred to Appendix D.

As for MIN-SUM-COST, we leverage operator $\otimes$ to prove a α-inapproximability result, for every $\alpha > 1$. Proof is in Appendix E.

D. *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, a slightly weaker computational complexity assumption, namely that not all problems in NP can be solved in "quasi-polynomial time", can lead to an even worse (i.e., higher) inapproximability factor: both MIN-ECMP-CONGESTION and MAX-ECMP-FLOW are hard to approximate within a non-constant factor that is "almost" a constant power of the size of the input instance. Again, this result is achieved via the repeated use of our gap-amplification technique (see, e.g., [6] for a similar approach).

Theorem 3.5: MIN-ECMP-CONGESTION and MAX-ECMP-FLOW cannot be approximated within a factor of $\left(\frac{3}{2}\right)^{(\log n)^{1-\epsilon}}$, where n is the number of edges of the input graph, unless NP is in quasi polynomial time.

Proof is in Appendix F.

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 [4]), or when traffic is split at a packet-level (instead of IP-flow-level, e.g., via Random Packet Spraying [13]), 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 paritioned 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 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 [3], [19], [32]). 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

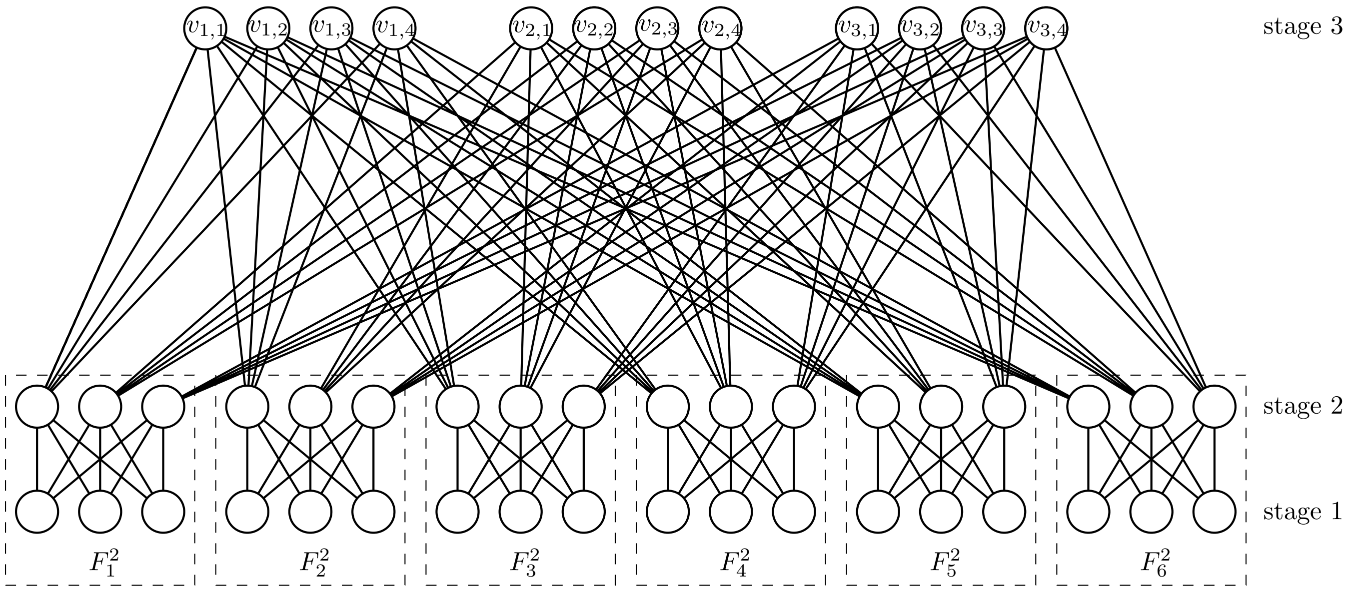

Fig. 5: A 3-FCN constructed by interconnecting four 2-FCNs.

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 [1].

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 consruction 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.

Proof of Theorem 7.1 is in Appendix G.

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 [4] and DevoFlow [11], 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 single shortest-path from s to t, such that the maximum load, i.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]. Proof is in Appendix H.

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. [3] and VL2 [19] advocate 3-FCNs as a datacenter topology, and Hedera [4] and DevoFlow [11] 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 in Appendix I). 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 [4] and can be regarded as a first step towards analyzing the provable guarantees of this family of heuristics.

Proof for 5-approximation. We introduce the following notation. Consider a 3-FCN F that contains k_r 2-FCN, 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. See Figure /reffig:clos-bounded. 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 linkof 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 exist. Let $s = b_i^j$ and $t = b_g^l$, with $i, g \in [k_b]$ and $j, l \in [k_r]$, where $[n] = 1, \ldots, n$. Observe that f_d is a lower bound for the optimal solution, i.e. $OPT \geq f_d$. It implies that $c(p_d) \geq 5 \cdot OPT \geq 5 f_d$. Let 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 $5 \cdot OPT$. Let 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 $5 \cdot OPT - f_d$ 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, i.e. $\mathcal{F}_v = \sum_{d'=((v,\cdot),\cdot)\in D} f_{d'} + \sum_{d'=((\cdot,v),\cdot)\in D} f_{d'}$. Hence we have that,

$$\mathcal{F}_{b_i^j} + \mathcal{F}_{b_g^l} \geq n_b(5 \cdot OPT) + n'_b(5 \cdot OPT - f_d) \geq$$

$$\geq 5 n_b OPT + n'_b(5 \cdot OPT - OPT) = 5 n_b OPT + 4 n'_b OPT$$

Let $\mathcal{F}_* = \max\{f^{b_i^j}, f^{b_g^l}\}$. We have that

$$2\mathcal{F}_* \geq 5 n_b OPT + 4 n'_b OPT \tag{1}$$

Consider now the following obvious lower bound for OPT

$$OPT \geq \frac{\mathcal{F}_{b_i^j}}{k_m}, OPT \geq \frac{\mathcal{F}_{b_g^l}}{k_m} \Rightarrow OPT \geq \frac{\mathcal{F}_*}{k_m} \tag{2}$$

In the first lower bound, we say that the total amount of flow originated from or directed to b_i^j must necessarily be splitted among its k_m edges that connect it to the vertices in the second stage. The same bound holds for b_g^l.

Combining (1) and (2), we obtain

$$k_m OPT \geq \frac{5 n_b OPT + 4 n'_b OPT}{2}$$

$$k_m \geq \frac{5 n_b + 4 n'_b}{2}$$

$$\frac{k_m}{2} \geq \frac{5}{4} n_b + n'_b \tag{3}$$

Let H be the set of indices h such that both (b_i^j, m_h^j) and (b_g^l, m_h^l) have congestion lower than or equal to $5 \cdot OPT - f_d$. By Equation (3), 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 $h \in H$, that has congestion less than $5 \cdot OPT - f_d$. This is a contradiction, since we assumed that such path does not exists. Hence, $j \neq l$. In this case, let n_t be the number of edges incident to any vertex t_x^h, with $h \in H$ and $1 \leq x \leq k_t$ and congestion at least $5 \cdot OPT$, and n'_t be the number of edges incident to any vertex t_x^h, with $h \in H$ and $1 \leq x \leq k_t$ and congestion between $5 \cdot OPT - f_d$ and $5 \cdot OPT$. Observe that each path (m_h^j, t_x^h, m_h^l) must have congestion at least $5 \cdot OPT - f_d$, 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 since we assumed that such path does not exists. Hence,

$$n_t + n'_t \geq |H| k_t \geq (k_m - n_b - n'_b) k_t \tag{4}$$

and we have that

$$\sum_{i\in[k_b]} \mathcal{F}_{b_i^j} + \sum_{i\in[k_b]} \mathcal{F}_{b_i^l} \geq n_t(5 \cdot OPT) + n'_t(5 \cdot OPT - f_d) \geq$$

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

where $\sum_{i\in[k_b]} \mathcal{F}_{b_i^j}$ ($\sum_{i\in[k_b]} \mathcal{F}_{b_i^l}$) is the sum of the flows originated from or directed to a vertex in the j-th (l-th) FCN. Let $\mathcal{F}_H = max\{\sum_{i\in[k_b]} \mathcal{F}_{b_i^j}, \sum_{i\in[k_b]} \mathcal{F}_{b_i^l}\}$. We have that

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

Consider now the following obvious lower bounds for OPT

$$OPT \geq \frac{\sum_{i\in[k_b]} \mathcal{F}_{b_i^j}}{k_t k_m}, OPT \geq \frac{\sum_{i\in[k_b]} \mathcal{F}_{b_i^l}}{k_t k_m} \Rightarrow$$

$$\Rightarrow OPT \geq \frac{\mathcal{F}_H}{k_t k_m} \tag{6}$$

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

In the first lower bound, we say that the total amount of flow originated from or directed to vertices in the j-th FCN must necessarily be splitted among its $k_m k_t$ edges that connect it to the last stage vertices. The same bound holds for the l-th FCN. Combining (5), and (6), we obtain

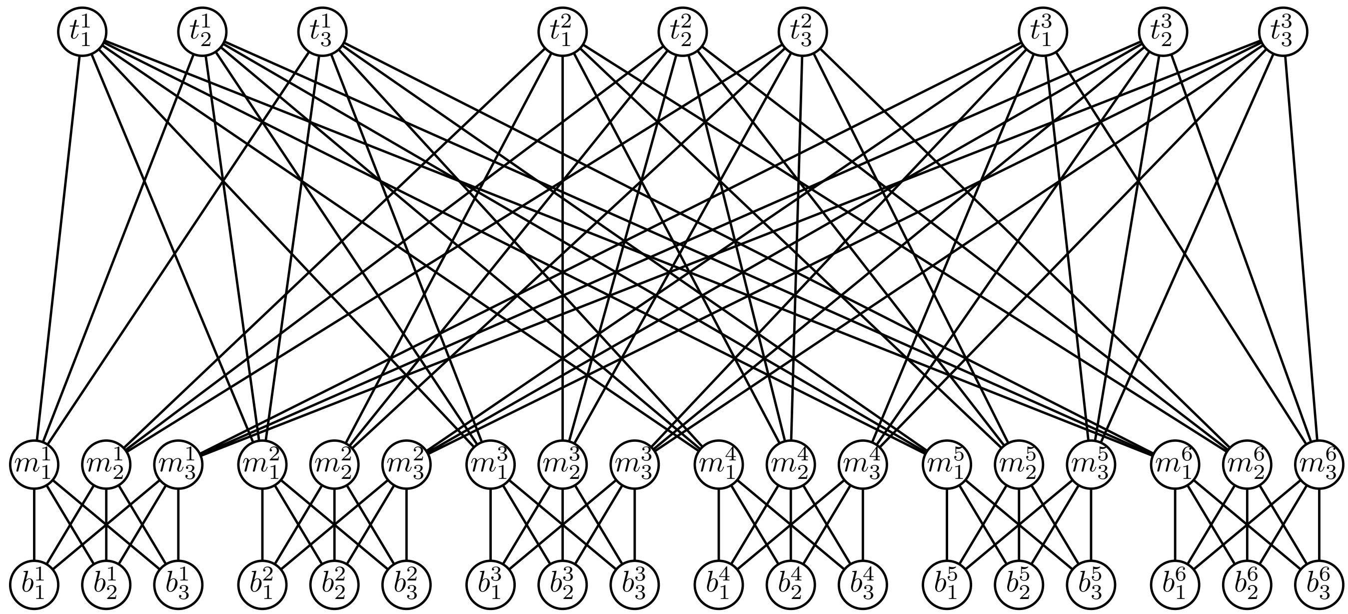

Fig. 6: A 3-FCN with $k_r = 6$, $k_b = 3$, $k_m = 3$, and $k_t = 3$.

$$k_t k_m OPT \geq \frac{OPT(5n_t + 4n'_t)}{2}$$

Using (4) and (3), we have that

$$2k_t k_m \geq 5n_t + 4n'_t = 4(n_t + n'_t) + n_t \geq 4k_t(k_m - n_b - n'_b) + n_t =$$

$$= 4k_t\left(k_m - \frac{5}{4}n_b - n'_b + \frac{1}{4}n_b\right) + n_t \geq 4k_t\left(\frac{k_m}{2} + \frac{n_b}{4}\right) + n_t$$

We have that

$$2k_m \geq 2\left(k_m + \frac{n_b}{4}\right) + \frac{n_t}{k_t}$$

$$0 \geq \frac{n_b}{2} + \frac{n_t}{k_t}$$

which 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 $c(p_d) < 5 \cdot OPT$. ■

Corollary 5.4: After $|\bar{D}|$ reroute operations, EQUILIBRIUM-ALGO approximates MCUF in 3-FCNs within a factor of 4, if all flows have equal size.

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 [5], [27] and [29] for more complete surveys.

TE with EMCP. We study TE with ECMP routing within the ("splittable flow") model of Fortz and Thorup [18]. Past work on optimizing ECMP routing mostly examined heuristic approaches (e.g., local search [18], branch-and-cut for mixed-integer linear programming [26], memetic [7] and genetic [14] algorithms) with no provable performance guarantees. [18] proves that mincongtitle 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 [12]. Topologies such as Clos networks [3] and generalized hypercubes [2], [20], [31] 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 [13] regarding packet-level traffic splitting in Clos networks, and also (ii) the experimental results in [4] 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". [32] 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 [32] 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 [9] for general networks. Recent work studies the routing of unsplittable flows in Clos datacenter networks [4], [11], [19] 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 [4] 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 [13], 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 [20], and MDCube [31]). What about other proposed datacenter topologies, e.g., random graphs ala Jellyfish [28]?
- **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 [4]) 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 bouded number of links. What can be said about the provable guarantees of TE with ECMP in this model?

REFERENCES

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

[2] Jung Ho Ahn, Nathan Binkert, Al Davis, Moray McLaren, and Robert S. Schreiber. Hyperx: topology, routing, and packaging of efficient large-scale networks. SC '09, 2009.

[3] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74, August 2008.

[4] Mohammad Al-fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *In Proc. NSDI*, 2010.

[5] Aysegul Altin, B. Fortz, and Hakan Umit. Oblivious ospf routing with weight optimization under polyhedral demand uncertainty. *Netw.*, 60(2):132–139, September 2012.

[6] Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, pcps and non-approximability – towards tight results, 1996.

[7] Luciana S. Buriol, Mauricio G. C. Resende, Celso C. Ribeiro, Mikkel, and Luciana S. Buriol Unicamp Campinas. A memetic algorithm for ospf routing, 2002.

[8] Zhiruo Cao, Zheng Wang, and Ellen W. Zegura. Performance of hashing-based schemes for internet load balancing. In *INFOCOM*, pages 332–341, 2000.

[9] Amit Chakrabarti, Chandra Chekuri, Anuptam Gupta, and Amit Kumar. Approximation algorithms for the unsplittable flow problem. In *In Proc. APPROX*, 2002.

[10] Cisco. Ospf design guide, 2011. http://www.cisco.com/image/gif/paws/7039/1.pdf.

[11] 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.*

[12] William Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[13] Advait Dixit, Pawan Prakash, and Ramana Rao Kompella. On the efficacy of fine-grained traffic splitting protocolsin data center networks. In *Proc. INFOCOM 2013*, 2013.

[14] 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*, 6:299–333, 2002.

[15] Bernard Fortz, Jennifer Rexford, and Mikkel Thorup. Traffic engineering with traditional ip routing protocols. *IEEE Communications Magazine*, 40:118–124, 2002.

[16] Bernard Fortz and Mikkel Thorup. Internet traffic engineering by optimizing ospf weights. In *INFOCOM*, pages 519–528, 2000.

[17] Bernard Fortz and Mikkel Thorup. Optimizing ospf/is-is weights in a changing world, 2002.

[18] Bernard Fortz and Mikkel Thorup. Increasing internet capacity using local search. *Comp. Opt. and Appl.*, 29(1):13–48, 2004.

[19] Albert G. Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. Vl2: a scalable and flexible data center network. *Commun. ACM*, 54(3):95–104, 2011.

[20] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. Bcube: A high performance, server-centric network architecture for modular data centers. In *In SIGCOMM*, 2009.

[21] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, July 2001.

[22] Ian Holyer. The np-completeness of edge-coloring. *SIAM J. Comput.*, 10(4):718–720, 1981.

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

[24] James R Lee and Assaf Naor. Embedding the diamond graph in lp and dimension reduction in l1. *Geometric & Functional Analysis*, 2004.

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

[26] Amandeep Parmar, Shabbir Ahmedy, and Joel Sokol. An integer programming approach to the ospf weight setting problem. Technical report, Georgia Institute of Technology, Oct 2006.

[27] Jennifer Rexford. Route optimization in ip networks. In *in Handbook of Optimization in Telecommunications, Springer Science + Business*. Kluwer Academic Publishers, 2006.

[28] Ankit Singla, Chi-Yao Hong, Lucian Popa, and P. Brighten Godfrey. Jellyfish: networking data centers randomly. NSDI'12, pages 17–17, Berkeley, CA, USA, 2012. USENIX Association.

[29] Peerapon Siripongwutikorn, Sujata Banerjee, and David Tipper. A survey of adaptive bandwidth control algorithms. *IEEE Communications Surveys and Tutorials*, 5(1):14–26, 2003.

[30] Ashwin Sridharan, Roch Guérin, and Christophe Diot. Achieving near-optimal traffic engineering solutions for current ospf/is-is networks. *IEEE/ACM Trans. Netw.*, 13(2):234–247, April 2005.

[31] Haitao Wu, Guohan Lu, Dan Li, Chuanxiong Guo, and Yongguang Zhang. Mdcube: a high performance network structure for modular data center interconnection. CoNEXT '09, 2009.

[32] Xin Yuan, Wickus Nienaber, Zhenhai Duan, and Rami Melhem. Oblivious routing for fat-tree based system area networks with uncertain traffic demands. *SIGMETRICS Perform. Eval. Rev.*, 35(1):337–348, June 2007.

Appendix A
Notation and Terminology

Let G be an undirected graph. We denote by $V(G)$ ($E(G)$) the set of vertices (edges) of G. We denote by $c_G(e)$ the capacity of edge $e \in E(G)$.

Reversibility. We now introduce an important property of a MAX-ECMP-FLOW instance that we will leverage to prove our inapproximability results. Let P be an optimization problem, $I = (H, s, t)$ be a MAX-ECMP-FLOW instance, where H is a directed acyclic graph from $s \in V(H)$ to $t \in V(H)$, and $P^*(I)$ be the value of the optimal solution of I. We say that a MAX-ECMP-FLOW instance I is *non-reversible* for P if $P^*(I) \geq P^*((H, t, s))$.

Appendix B
MAX-ECMP-DAG Constant Inapproximability

Refer to Appendix A for notation and terminology details.

Relation between MIN-ECMP-CONGESTION and MAX-ECMP-FLOW. The following theorem has been proved by Fortz and Thorup [18].

Theorem 2.1: Given a MIN-ECMP-CONGESTION instance I, with unit flow demands in D, distinguishing between the following two scenarios is NP-Hard:

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

where $OPT_{MC}(I)$ is the value of the optimal solution for I.

We prove the following lemma, based on Theorem 2.1.

Lemma 2.2: Given a MIN-ECMP-CONGESTION instance I with a single source-target pair, distinguishing between the following two scenarios is NP-Hard:

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

Proof: Let $I = (G, D)$ be an MEC instance such that OPT_{MC} is either 1 or $\frac{3}{2}$, where $F = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ is the set of source-target pair of vertices of G with $D_{s_i,t_i} > 0$. 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 $D_{s,t} = |F|$, $D_{x,y} = 0$ 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 1 and all the remaing edges of both binary trees to infinite. We now show that $OPT_{MC}((G, D)) = OPT_{MC}((G', D'))$. It is easy to see that $OPT_{MC}((G, D)) \geq OPT_{MC}((G', D'))$. In fact, if (i) flow demand $D_{s,t}$ is splitted 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 $OPT_{MC}((G, D))$. By observing that an unequal splitting through the binary tree from s to vertices $s_1, \ldots, s_k$ causes a congestion of 2, the lemma easily follows. ■

An instance of MIN-ECMP-CONGESTION with a single source-target unit flow demand is denoted by $I = (G, s, t)$. Similarly, an instance of MAX-ECMP-FLOW with a single source-target flow demand is denoted by $I = (G, s, t)$.

Lemma 2.3: 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 sent from s to t by a factor of $\frac{1}{OPT_{MC}(I)}$, each edge will have congestion at most 1. Hence, $OPT_{MF}(I) \geq \frac{1}{OPT_{MC}(I)}$. Viceversa, 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 $\frac{1}{OPT_{MF(I)}}$, each edge will have congestion at most $\frac{1}{OPT_{MF(I)}}$ and a unit of flow will be routed from s to t. Hence, $OPT_{MC}(I) \leq \frac{1}{OPT_{MF}(I)}$. ■

Corollary 2.4: Given a MAX-ECMP-FLOW instance I with a single source-target pair, distinguishing between the following two scenarios is NP-Hard:

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

Proof: It easily follows by Lemma 2.3 and Theorem 2.2. ■

We say that a flow assignemnt f on a graph G is *realized* by a weight assignment w, if setting link weights of G as in w, flows are routed according to f. In the proof of Theorem 2.1, it was observed that, given an input instance $I = (G, D)$ of MEC, it is possible to orient every edge of G in such a way that at least one among the optimal flow assignment realized by an optimal weight assignment is "compliant" with such orientation, i.e., for every edge $(x, y) \in E(G)$, if a flow is routed from x to y, then (x, y) is oriented from x to y. Also, authors observed that at least a optimal flow assignment, not necessarily realized by a link weights assignment, where flows are splitted equally, is no better than the one obtainable with an optimal link weights assignment. Hence, considering an instance (G, s, t) of MIN-ECMP-CONGESTION with a single source-target unit flow from s to t, we have that $OPT_{MF}((G, s, t)) = OPT((G, s, t))$. The following lemma guarantees that every flow assignment with a single source-target flow demand can be realized by a weight assignment.

Assignment of weight links and directed acyclic graphs. For a weight function w on the edges of G, consider the ECMP flow associated with w, and let $\mathcal{A}(w)$ denote the oriented subgraph of G containing the edges which have nonzero flow, and directed according to the direction of the flow. Since ECMP routes flows along shortest-paths, $\mathcal{A}(w)$ is a DAG with a source at s and a sink at t. We call such a DAG an (s, t)-DAG in $I = (G, s, t)$. Note that given $\mathcal{A}(w)$, the associated ECMP flow can be regenerated even without knowing w.

The following lemma shows that for every (s, t)-DAG H there are weights w such that $\mathcal{A}(w) = H$. This means that optimizing over weights is equivalent to optimizing over (s, t)-DAG, which justify our reduction to the MAX-ECMP-DAG problem.

Lemma 2.5: For any arbitrary (s, t)-DAG A of (G, s, t), there exists an assignment w of the edge weights of G such that $\mathcal{A}(w)$ is equal to A.

Proof: We denote by $out(v, A)$ the set of vertices of A that have an ingoing edge from vertex v. and by $sp(v, t, w)$ the length of the shortest-paths from vertex v to vertex t of G according to a link weight assignment w. Consider a topological order $(v_1, \ldots, v_n)$ of the vertices of A, where $v_1 = t$ and $v_n = s$. An assignment of the weights to the edges of A that satisfies the claim of the lemma is computed by the following procedure that processes vertices from v_2 to v_n. For each vertex v_i, with $i = 2, \ldots, n$, let M be the length of the longest shortest path from any neighbor of $v_i \in out(v_i, A)$,i.e., $M = \max_{v_l \in out(v_i, A)}\{sp(v_l, t, w)\}$. For each vertex $v_k \in out(v_i, G)$, we set $w((v_k, v_i)) = M + 1 - sp(v_k, t, w)$. This guarantees that all the edges in $out(v_i, A)$ belong to at least one shortest path from v_i to t. Observe that, since vertices are processed in topological order, we have that $l < i$ and $k < i$, which implies that both $sp(v_l, t, w)$ and $sp(v_k, t, w)$ are defined when vertex v_i is processed. Observe that this assignment implies that the shortest path in G from s to t is at most $|E(G)|$. For this reason, for each edge $e \in E(G) \setminus E(A)$, we set $w(e) = |E(G)| + 1$. This guarantees that every shortest path between s and t does not pass through an edge that is not contained in A. ∎

This lemma and the previous consideration about the relation between MAX-ECMP-FLOW and MAX-ECMP-DAG leads to our theorem. Recall that every MAX-ECMP-DAG instance is a directed acyclic graph with single source and target vertex.

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.

APPENDIX C
⊗ AMPLIFICATION

Lemma 3.3. Let I be an instance of MAX-ECMP-DAG. *$OPT(\otimes^k I) = (OPT(I))^{k+1}$ 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 $\otimes^0 I$.

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 (i) $OPT(I_{k+1}) = OPT(I)^{k+2}$. 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 by a DAG H_e in I_{k+1}, where the capacity of each edge of H_e is multiplied by $c_{I_k}(e)$. Consider a solution $\bar{H}_{k+1}$ where each vertex of I_{k+1} 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 $\bar{H}_{k+1}$. Now, we show that in each subgraph H_e if we split traffic as $\bar{H}_0$ does in I we can route through H_e a flow that is $OPT(I_0)$ times larger than $c_{I_k}(e)$. 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 $E(\bar{H}_{i+1})$. Therefore, the maximum flow in I_{k+1} is $OPT(I) \cdot OPT(I_k) = OPT(I) \cdot OPT(I)^{k+1} = OPT(I)^{k+2}$, 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 $OPT(I_k) = OPT(I)^{k+1}$. ∎

APPENDIX D
RELATING MAX-ECMP-DAG TO MAX-ECMP-FLOW AND MIN-ECMP-CONGESTION

Refer to Appendix A for basic notation and terminology details. Given an instance H_0 of MAX-ECMP-DAG, let G_k be an undirected copy of $\otimes^k H_0$. 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.

Enforcing non-reversibility. Consider $I_0 = (G_0, s, t)$ such that $OPT_{MF}(I_0)$ is either 1 or $\frac{2}{3}$. If G_0 is non-reversible, it is possible to transform G_0 in a non-reversible instance with the following construction. Let r be the ratio between the value of the optimal solution for (G_0, s, t) and (G_0, t, s). Since I_0 is non-reversible, $r < 1$. Add a new source vertex s' into $V(G_0)$ and connect it through a path $(s', v_1, v_2.v_3, v_4, s)$ to s, with infinite capacity on its edges. Then, connect s' to each vertex v_1, v_2, v_3, v_4 with an edge of capacity $\frac{1}{4}$. Let s' be the new source vertex of I_0. Observe that, by construction, the maximum flow from s to s' is $\frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32}$, while the maximum flow from s' to s is 1 since s' can send a unit of flow to v_1, v_2, v_3, v_4, respectively. Hence, $OPT_{MF}(G_0, s', t) = \min\{OPT_{MF}(G_0, s', s), OPT_{MF}(G_0, s, t)\} = \min\{1, OPT_{MF}(G_0, s, t)\} = OPT_{MF}(G_0, s, t) \geq \frac{2}{3} \geq \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} = OPT_{MF}(G_0, s, s') \geq OPT_{MF}(G_0, t, s')$, which means that I_0 is now non-reversible.

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 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. We say that a (s, t)-flow 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 $(x, y) \in E(H_0)$.

Lemma 4.1: Suppose that in at least one optimal solution for (G_0, s, t) the (s, t)-flow is compliant with H_0. Then, $OPT(H_k) = OPT_{MF}(I_k)$.

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 2.5 we know that $OPT_{MF}(I_k) = OPT(H_k) = OPT(H_0)^{k+1}$ and $OPT_{MF}(I_{k+1})$ is at least $OPT(H_{k+1}) = OPT(H_0)^{k+2}$. We want to show that $OPT_{MF}(I_{k+1}) \leq OPT(H_0)^{k+2}$. 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 $OPT_{MF}(H_e, t_e, s_e) \leq OPT_{MF}(H_e, s_e, t_e) = OPT_{MF}(H_0, s, t) \cdot c_{I_k}(e)$, which means that, for each edge e of H_k, we can route at least a flow $OPT_{MF}(I_0)$ times smaller than in H_e. Hence, solution $\bar{H}_k$, induces a maximum flow through H_k of at least $\frac{OPT_{MF}(I_{k+1})}{OPT_{MF}(I_0)} > \frac{OPT_{MF}(I_0)^{k+2}}{OPT_{MF}(I_0)} = OPT_{MF}(I_0)^{k+1}$ units, which is a contradiction, since, by induction hypothesis, we have that $OPT_{MF}(I_k) = OPT_{MF}(I_0)^{k+1}$. ■

By Lemma 2.3. we have the following corollary.

Corollary 4.2: $OPT(H_k) = \frac{1}{OPT_{MC}(I_k)}$.

We can prove our main lemma.

Lemma 3.4. 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 constrct a α-approximation algorithm for MAX-ECMP-DAG as follows. Let H_0 be MAX-ECMP-DAG instance such that its undirected copy G_0 is non-reversible and $OPT(H_0)$ is either 1 or $\frac{2}{3}$. Let s and t be the source and sink vertices of H_0. 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 4.1, 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 instaces with optimal value 1 or $\frac{2}{3}$, which is a contradiction to Theorem 3.2.

By Lemma 2.3, the same result also holds for MIN-ECMP-CONGESTION. ■

APPENDIX E
SUM OF LINK COSTS INAPPROXIMABILITY

Refer to Appendix A for notation and terminology details. We now consider the MIN-SUM-COST problem closely. As observed in [16], minimizing the maximum congested edge is a suitable measure to be minimized in the case a network administrator wants to ensure that *no* packet gets sent across overloaded arcs (e.g., because QoS requirements). Hence, depending on the context, the inapproximability result of Theorem 3.1 may be considered over-pessimistic. In fact, even if just one flow, out of hundred of thousand of flows, is routed through a congested path, the value of the most congested link may result in a misleading measure of the congestion of the entire network. In this case, a sum of link costs can be adopted as the optimization function to be minimized. Past works [15], [16], [17] shown that the link cost increases progressively as the congestion approaches 1, and explodes when we go above 1. We model this behaviour by an exponential function $\phi(x) = 2^x - 1$, where x is a measure of the congestion of the link.

We again exploit our construction technique based on operator $\otimes$ and we show that routing flows with ECMP remains inapproximable even with a sum of link costs objective function:

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

We first introduce and study a problem called MIN-CONGESTED-EDGES, where the goal is to minimize the number of edges that has congestion at least $\frac{3}{2}$. We then show how to leverage our $\otimes$ construction technique in order to amplify the gap between two different class of instances of MIN-CONGESTED-EDGES. Finally, we relate in Theorem 5.4 MIN-CONGESTED-EDGES to MIN-SUM-COST showing that the the latter is not approximable within any constant.

MIN-CONGESTED-EDGES problem. In this problem, an instance I is graph G where a vertex $s \in V(G)$ wants to send a flow of f units to a vertex $T \in V(G)$. The goal is to minimize the number of edges that has congestion at least $\frac{3}{2}$. A solution of the problem is given by an (s,t)-DAG.

In the Fortz and Thorup reduction construction from 3-SAT [18], a formula F with n variables is transformed into an instance $I = ((G,s,t),\cdot)$ of MIN-ECMP-CONGESTION such that, if an 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. 5.1) to prove that also in a slightly modified Fortz and Thorup construction, at least a certain amount of edges must be congested (Lemma 5.2).

Theorem 5.1: [21]. 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 Thm. 5.1 and on a straightforward modification of the Fortz and Thorup construction.

Lemma 5.2: There exists a constant $\alpha > 1$ such that, given a congestion threshold $C = \frac{3}{2}$, it is NP-Hard to approximatate 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 $p > 0$ 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 $k > 0$ 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 $k > 0$ 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 compliant with H. We denote by G_k the undirected copy of $\otimes^k H$ and by $I_k = (G_k,s,t)$.

Lemma 5.3: 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 (s,t)-DAG $\bar{A}$ of I_{k+1} such that less than $p^{k+2}|E(G_{k+1})|$ edges have congestion at least C^{k+2}. We now construct an (s,t)-DAG A_k of I_k from $\bar{A}$ exactly as we did in the proof of Lemma 3.3.

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 at least a fraction p of 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 $C_e \geq C f_e$, where f_e is the amount of flow routed from s_e to t_e in G^{k+1}. Since an (s,t)-DAG A_k implies that edge $e \in E(G_k)$ is also traversed by a flow f_e, its congestion is $\frac{C_e}{C}$. By a simple counting argument, there 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 A_k induces a flow through G_k 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 5.2. We then leverage our construction technique based on operator $\otimes$ on these instances. As a consequence, by Lemma 4.2 and Lemma 5.3, the gap between the optimal sum of link costs can be arbitrary high.

Theorem 5.4: It is NP-Hard to approximate the MIN-SUM-COST problem within any constant factor.

Proof: Suppose that there exists a α-approximation algorithm for a certain constant α. Let $I = (G,s,t)$ be an instance of MIN-CONGESTED-EDGES, where I is a non-reversible SINGLE-SOURCE-TARGET instance, in its optimal solution either (i) all edges are congested at most 1 or (ii) at least a fraction p of its edges have congesttion at least C, and 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 a solution of the MIN-SUM-COST problem on an instance constructed with operator $\otimes$.

In case (i), by Lemma 4.2, each edge of G_k in the optimal solution of I^k have 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 5.3, 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 greater 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 5.2. ■

APPENDIX F
NON-CONSTANT (ALMOST POLYNOMIAL) INAPPROXIMABILITY FACTORS

Refer to Appendix A for notation and terminology details. 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 one can push further the technique of Lemma 3.3. We can show that MAX-ECMP-FLOW is hard to approximate even within a non-constant factor which is almost a constant power of the size of the input instance. This hardness result is obtained by a rather standard computation (see [6]) which we explain below. 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 6.1: 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 hardnes-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 6.1 let us start with the following technical lemma.

Lemma 6.2: 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 the Theorem.

Proof of Theorem 6.1: We repeat the construction as in Lemma 3.3, 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 6.2 we have that

$$|E(\otimes^k I)| \leq |E(I_0)|^{k+2}, \tag{7}$$

and thus $\otimes^k I$ can be constructed from I_0 in quasi-polynomial time. By Lemma 3.3, 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 7, 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 statementof the theorem. ■

APPENDIX G
TE WITH ECMP IS NP-HARD FOR HYPERCUBES

Refer to Appendix A for notation and terminology details.

Theorem 7.1: Computing the optimal flow in hypercubes with respect to MIN-ECMP-CONGESTION is NP-Hard.

Proof: we leverage instances used in Theorem 2.1 to prove the hardness result for hypercube topologies. In particular, we consider an instance $I = (G, s, t)$ such that either $OPT_{MC}(I) = 1$ or $OPT_{MC}(I) = \frac{3}{2}$. We "embed" I into an hypercube H, with a logarithmic dimension w.r.t. to the size of I, and we carefully construct a demand matrix D for vertices in $V(H)$ in such a way that $OPT_{MC}(H, D) = 1$ iff $OPT_{MC}(I) = 1$, where $OPT_{MC}(H, D) = 1$ is the value of an optimal weight assignment for a graph H with demand matrix D.

Embedding sketch idea. Let $I = ((G, s, t), f)$ be an instance of MIN-ECMP-CONGESTION, where a vertex s of G wants to sends a flow of f units to avertex t of G. Consider an hypercube H that contains a subgraph G' that it is a subdivision of G, i.e., a *subdivision* G' of a graph G can be obtained from G by replacing each edge of G with a single simple path. We can show that such hypercube exists and has size polynomial w.r.t. the maximum degree of a vertex of G and the size of G. We then construct a demand matrix among vertices of H. We add a flow between the endpoints of each edge $e \in E(H)$ in such a way that we saturate the capacity of e only if e is not in $E(G')$. In order to enforce capacity constraints of edges of G to paths in H, for each path p of G' that corresponds to an edge e of G, for each edge e' of

p, we assign a certain flow between the endpoints of e' that limits the capacity of that path. Namely, the higher the value of the capacity of e, the lower the size of the flow that we assign between the endpoints of e'. These flows are used to force a flow from s to t to flow exactly through G'. We then refine are mapping by removing "chords" from the subgraph induced by vertice of G'. Since G' is a subdivision of G and the available capacity of each edge of G' is properly scaled by these extra-flows, we can prove that if $OPT_{MC}(I) = 1$, then $OPT_{MC}(H, D) = 1$. Otherwise, if $OPT_{MC}(I) > 1$, then $OPT_{MC}(H, D) > 1$. Since MIN-ECMP-CONGESTION is NP-Hard even in the case G has degree at most 3 and $OPT_{MC}(I)$ is either 1 or $\frac{3}{2}$ [18], then also MIN-ECMP-CONGESTION is NP-Hard even if we restrict our attention to hypercubes.

We now show how to find a subgraph G' of an hypercube such that G' is a subdivision of G.

Embedding a graph instance into an hypercube. Consider an instance (G, D) of MIN-ECMP-CONGESTION, where D contains only a flow demand f from s to t. We first map G into a k-hypercube H, with $k > 0$. Let ϕ be an injective function that maps each vertex v of G to a vertex $\phi(v)$ of H, each edge $e = (v, u)$ of I to a simple path $\phi(e)$ of H from $\phi(v)$ to $\phi(u)$. Let G' be the subgraph of H such that $e \in E(G')$ iff there exists an edge $e' \in E(G)$ such that $\phi(e')$ traverses e. We say that ϕ is an *embedding* of G into H if G' is a subdivision of G. In other words, for each pair of edges e_1 and e_2 of G, paths $\phi(e_1)$ and $\phi(e_2)$ are internal-vertex-disjoint.

We construct H from G with the following recursive procedure. Let $M = \max\{|V(G)|2^{\Delta(G)-1}, 2|E(G)|\}$, where $\Delta(G)$ is the degree of the vertex with maximum degree, and $d = 2\lceil \lg_2 M \rceil$. We first construct a d-hypercube H using the following notation to denote its vertices. H contains 2^d vertices $v_{(x,y)}$, with $0 \le x \le 2^{\frac{d}{2}} - 1$ and $0 \le y \le 2^{\frac{d}{2}} - 1$. There exists an edge between two vertices v_{xy} and v_{wz} iff there exists a $n \ge 0$ such that either $(x = w) \wedge (|z - y| = 2^n) \wedge \left(\lfloor \frac{y}{2^{n+1}} \rfloor = \lfloor \frac{z}{2^{n+1}} \rfloor\right)$, or $(y = z) \wedge (|x - w| = 2^n) \wedge \left(\lfloor \frac{x}{2^{n+1}} \rfloor = \lfloor \frac{w}{2^{n+1}} \rfloor\right)$.

We now create a mapping ϕ from G to H. Let $u_0, \ldots, u_n$ be all the vertices of G. Let $\phi(u_i) = v_{(0,2^{\Delta(G)-1}i)}$. Let A be the set of edges of G that has been mapped to a path of H. Initially, $A = \emptyset$. Each vertex v order its incident edges into a sequence E_v. For each edge $e = (a, b) \in E(G)$, we compute a path p_i from $\phi(a) = v_{(0,y)}$ to $\phi(b) = v_{(0,z)}$, where $y = \phi(a)$ and $z = \phi(b)$, as a concatenation of 5 paths p_1, p_2, p_3, p_4 and p_5. Suppose e is the i-th (j-th) edge in E_a (E_b). If $i \ne 1$, $p_1 = (v_{(0,y)}, v_{(0,y+2^{i-1})})$, otherwise $p_1 = (v_{(0,y)})$. If $j \ne 1$, $p_5 = (v_{(0,z)}, v_{(0,z+2^{j-1})})$, otherwise $p_5 = (v_{(0,z)})$. p_2 is a shortest path from $v_{(0,y+2^{i-1})}$ to $v_{(2|A|+1,y+2^{i-1})}$. p_4 is a shortest path from $v_{(0,z+2^{j-1})}$ to $v_{(2|A|+1,z+2^{j-1})}$. p_3 is a shortest path from $v_{(2|A|+1,y+2^{i-1})}$ to $v_{(2|A|+1,z+2^{j-1})}$. Add e into A.

See an example of embedding a clique K_4 of size 4 into an hypercube H of dimension $d = 2\lceil \lg_2 M \rceil = \lceil \lg_2(\max\{|V(G)|2^{\Delta(G)-1}, 2|E(G)|\}) \rceil = 2\lceil \lg_2(\max\{4 \cdot 2^{3-1}, 12\}) \rceil = 2\lceil \lg_2(12) \rceil = 8$ in Fig. 7. Each vertex $v_{(x,y)}$ of H is represented as a circle in column x and row y.

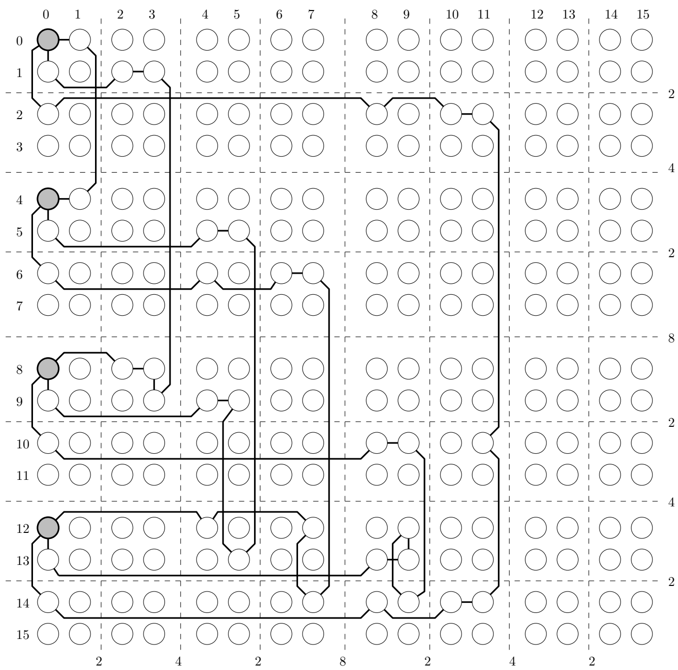

Fig. 7: A clique with 4 vertices embedded in a 8-hypercube. Vertices of the clique are mapped to vertices $v_{0,0}, v_{0,4}, v_{0,8}$ and $v_{0,12}$, depicted as gray vertices. Dashed lines are depicted in order to ease the readability of this figure. Let l be the highest number of a dashed line intersected by an edge $(v_{x,y}, x_{x,z})$ $((v_{x,y}, x_{w,y}))$. Then, it must hold that $|y-z| = l$ $(|x-w| = l)$.

.

Edges of the hypercube are not depicted. In order to better understand where an edge between two vertices of H exists, we drew several dashed lines, each of them with a number beside it. An edge of the hypercube is such that, if it intersect any dashed line, then its two endpoints have a diffference in one of their coordinate that is exactly equal to the highest number associated to any of the intersected dashed lines. Let u_0, u_1, u_2, u_3 be all the vertices of K_4. We have that $\phi(u_0) = v_{(0,0)}, \phi(u_1) = v_{(0,4)}, \phi(u_2) = v_{(0,8)}$, and $\phi(u_3) = v_{(0,12)}$. These vertices are colored gray in the picture. Let (u_0, u_1) be the first edge of K_4 to be analyzed and let (u_0, u_1) be the first edge in both E_{u_0} and E_{u_1}. We have that path $p_1 = (v_{(0,0)})$, path $p_2 = (v_{(0,0)}, v_{(1,0)})$, path $p_3 = (v_{(1,0)}, v_{(1,4)})$, path $p_4 = (v_{(1,4)}, v_{(0,4)})$, and $p_5 = (v_{(0,4)})$. Now, consider edge (u_0, u_2) and suppose that it is the second (first) edge in E_{u_0} (E_{u_2}). We have that path $p_1 = (v_{(0,0)}, v_{(0,1)})$, path $p_2 = (v_{(0,1)}, v_{(2,1)}, v_{(3,1)})$, path $p_3 = (v_{(3,1)}, v_{(3,9)}, v_{(3,8)})$, path $p_4 = (v_{(3,8)}, v_{(2,8)}, v_{(0,8)})$, and $p_5 = (v_{(0,8)})$. Intuitively, for each edge (x, y), path p_1 (p_5) is used to connect $\phi(x)$ ($\phi(y)$) to a vertex z_x (z_y) in the first column in the first "available" row below $\phi(x)$ ($\phi(y)$). Path p_1 and p_5 may be empty in the case (x, y) is the first edge in E_x and E_y, respectively. Then, path p_2 (p_4) is used to connect z_x (z_y) to a vertex w_x (w_y) of a column with index $2|A|+1$. Observe that, since p_2 and p_4 are shortest paths, they never change row. Also, observe that each of these paths has only one vertex that lies on a even column, namely on column $2|A|+1$. Finally, path p_3 is used to interconnect these two vertices w_x and w_y. Also in this case, path p_3 never leaves column $2|A|+1$. Because of these properties, it is easy to see that for each pair of edges e_1 and e_2 of G, paths $\phi(e_1)$ and $\phi(e_2)$ are internally vertex-disjoint.

Assignign flow demands in H. We construct set D from $((G,s,t),f)$, H, and ϕ. For each edge $e \in E(G)$, for each edge (x,y) of $\phi(e)$, add flow $((x,y), \frac{2}{5}\left(\frac{c_{max}-c_G((x,y))}{4c_{max}}+1\right)$ and $((y,x), \frac{2}{5}\left(\frac{c_{max}-c_G((x,y))}{4c_{max}}+1\right)$ into D. For each edge $e' \in H$ that has no flow assigned, add flows $((x,y),\frac{1}{2})$ and $((y,x),\frac{1}{2})$ into D. Finally, add a flow $((\phi(s),\phi(t)),\frac{1}{5c_{max}})$.

Removing chords from G'. Consider the subgraph of H induced by vertices of G', where $V(G') = \{v \in V(H') | \exists e \in E(G)$ such that $\phi(e)$ passes through $v\}$ and $E(G') = \{e' \in E(H') | \exists e \in E(G)$ such that $\phi(e)$ traverses $e\}\}$. By the above construction, G' may contain edges that are not in $E(G')$. We call these edge "chords". We now show a procedure that, given a mapping ϕ of a graph G into a k-hypercube H, produces a new mapping ϕ' from G' to a $2k$-hypercube H' such that the subgraph of H' induced by vertices in $\{v \in V(H') | \exists e \in E(G')$ such that $\phi'(e)$ passes through $v\}$ is chordless. The construction works as follow. Map each vertex $x = (x_0,\dots,x_k)$ of H, where $x \in V(G')$, to vertex $\phi'(x) = (q_x,q_x) = (x_0,\dots,x_k,x_0,\dots,x_k)$ of H. Consider each edge $e = (x,y) = ((x_0,\dots,x_k),(y_0,\dots,y_k))$ of H, where $(x,y) \in E(G')$. Since (x,y) is an edge of an hypercube, x and y differs in exactly one coordinate i. Map (x,y) to a path $\phi'(e) = (\phi'(x),\bar{x}^i,\phi'(y))$, where $\bar{x} = (q_x^i,q_x) = (x_0,\dots,x_{i-1},y_i,x_{i+1},\dots,x_k,x_0,\dots,x_k)$, where q_x^i is obtained by flipping the i-th coordinate of q_x. Let G'' be a subgraph of H', such that $V(G'') = \{v \in V(H') | \exists u \in V(G')$ s.t. $\phi'(u) = v\}$ and $E(G') = \{e \in E(H') | \exists e' \in E(G')$ s.t. $\phi'(e')$ traverses $e\}$.

Lemma 7.2: G'' is chordless.

Proof: Suppose, by contradiction, that G'' is not chordless. Then, there exists at least a pair of vertices x and y of G'' that are not adjacent in G'' and are adjacent in H'. Let i be the coordinate in which x and y differs by one element. We have two cases. If there exist two vertices a and b of G' such that $\phi'(a)=x$ and $\phi'(b)$, then, by construction of G'', since a and b differs in one coordinate, we have that x and y must differ in two coordinates. This is a contradiction since (x,y) cannot be a chord. Otherwise, if such two vertices does not exists, w.l.o.g., let x be a vertex such that there does not exist a vertex a of G' with $\phi'(a)=x$. Observe that, by construction of ϕ', each vertex $z=(q_z,q_z)$ of $\phi'(G)$ has coordinate either (q_z,q_z) or (q_z^j,q_z), with $j=1,\dots,k$. Hence, x has coordinates (q_x^j,q_x), with $j=1,\dots,k$. x and, since y is a neighbor of x and it is a vertex of G'', it must have coordinate either (q_x,q_x) or (q_x^i,q_x^i). This leads to a contradiction since, by construction of G'', both $((q_x,q_x),(q_x^i,q_x))$ and $((q_x^i,q_x),(q_x^i,q_x^i))$ are edges of G''. ■

Assignign flow demands in H'. We construct set D' from (H,D) and ϕ' as follows. For each flow demand D_{xy} in D, with $x \neq s$ and $y \neq t$, for each edge $e' = (x',y')$ traversed by $\phi'((x,y))$, add a flow demand $D_{x'y'} = D_{xy}$ into D'. For each edge $e' = (x',y') \in E(H')$ such that there does not exists an edge e of G' such that $\phi'(e)$ traverses e', add flows $((x',y'),\frac{1}{2})$ and $((y',x'),\frac{1}{2})$ into D. Finally, add a flow demand $D'\phi'(s),\phi'(t) = D_{st}$ into D'.

Proving optimal solution for (H',D').

Lemma 7.3: If $OPT_{MC}(I)=1$, then $OPT_{MC}(H',D')=1$.

Proof: We first compute a weight assignment of (H',D') from a weight assignment of (I,f) that has congestion at most 1. We first map each flow (x,y) in D' to an (x,y)-DAG $\sigma((x,y))$. Then, we compute weight links from this set of DAG. Suppose $OPT_{MC}(I)=1$. By construction of D', each flow $((x,y),1) \in D'$, with $x \neq \phi'(\phi(s))$ and $y \neq \phi'(\phi(t))$, is such that $(x,y) \in E(H')$. Let $\sigma((x,y)) = A_{xy}$, where A_{xy} is a (x,y)-DAG that consists of a single edge (x,y). Consider an optimal (s,t)-DAG A_I of (I,f). We construct an $(\phi'(\phi(s)),\phi'(\phi(t)))$-DAG A_{st} from A_I. For each edge $e \in E(A_I)$, add directed path $\phi'(\phi(e))$ into $E(A_{st})$. Finally, let $\sigma((\phi'(\phi(s)),\phi'(\phi(t)))) = A_{st}$. By construction of (H',D'), it is trivial to see that this solution has congestion equal to 1. Consider an edge $(x,y) \notin A_{st}$. By construction of (H',D') and σ, only two saturating flows $((x,y),\frac{1}{2})$ and $((y,x),\frac{1}{2})$ are routed through (x,y), which implies that (x,y) has congestion equal to 1. Consider now an edge $(x,y) \in E(A_{st})$. By construction of (H',D') and σ, we have that (x,y) is traversed by a fraction of the flow from $\phi'(\phi(s))$ to $\phi'(\phi(t))$ and, a saturating flow $\left((y,x),\frac{2}{5}\left(\frac{c_{max}-c_G((u,v))}{4c_{max}}+1\right)\right)$, where (u,v) is the edge of A_I and edge e' of $\phi((u,v))$ is such that (x,y) is an edge of $\phi'(e')$, and a saturating flow $\left((y,x),\frac{2}{5}\left(\frac{c_{max}-c_G((u,v))}{4c_{max}}+1\right)\right)$. Observe that, if a fraction q of the unit flow from s to t is routed through (u,v) in G, then, by construction of A_{st}, also a fraction q of the flow from $\phi'(\phi(s))$ to $\phi'(\phi(t))$ is routed through (x,y). Hence, we have that the amount of flow routed through (x,y) is

$$2 \cdot \frac{2}{5}\left(\frac{c_{max}-c_G((u,v))}{4c_{max}}+1\right)+q\frac{1}{5c_{max}} =$$

$$= \frac{4}{5}\left(\frac{c_{max}-c_G((u,v))}{4c_{max}}+1\right)+q\frac{1}{5c_{max}} =$$

$$= \frac{1}{5}\left(\frac{c_{max}-c_G((u,v))+4c_{max}+q}{c_{max}}\right) =$$

$$= \frac{1}{5}\left(\frac{5c_{max}-c_G((u,v))+q}{c_{max}}\right) =$$

$$= 1-\frac{c_G((u,v))-q}{c_{max}} \leq 1, \text{ since } q \leq c_G((u,v)).$$

We now show how to set link weights in H' in order to obtain a flow assignment where flow is routed according to

σ. For each saturating flow (x, y) of H' such that $(x, y) \notin E(G'')$, we set a very large weight $W >> 1$ to edge (x, y). Since G'' is chordless, each of these flow is routed through (x, y). Let $\bar{\sigma} = \sigma(\phi'(\phi(s)), \phi'(\phi(t)))$. We set link weights in $\bar{\sigma}$ using Lemma 2.5. Hence, flow demand $(\phi'(\phi(s)), \phi'(\phi(t)))$ is routed through $\bar{\sigma}$ and each saturating flow (x, y) of H' such that $(x, y) \in E(G'')$ is routed exactly through (x, y). This concludes the proof of the lemma. ■

Lemma 7.4: If $OPT_{MC}(I) > 1$ and G has maximum degree 3, then $OPT_{MC}(H', D') > 1$.

Proof: Suppose, by contradiction, that there exists an assignment of the link weights such that $mc^*(H', D') \leq 1$. Among these optimal assignments, consider the one that has the higher number of saturating flows routed through the edge that interconnects their source and target vertices. For each flow $((x, y), \cdot) \in D'$, let A_{xy} be the (x, y)-DAG where the flow is routed through. We show that for each saturating flow $((x, y), \frac{1}{2})$, where $(x, y) \in E(H')$, we have that A_{xy} consists of a single edge (x, y). Suppose, by contradiction, that it is not true. Observe that, if A_{xy} contains an edge (x', y'), then $A_{x'y'} \subset A_{xy}$. It implies that there must exists a saturating flow $f^* = ((x, y), \cdot)$ such that A_{xy} consists of at least a directed path p different from (x, y). Observe that A_{yx} contains an edge (u, v) iff A_{xy} contains an edge (v, u). Let $A*$ be such (x, y)-DAG of a saturating flow $((x, y), \cdot)$ such that it does not consist of a single edge (x, y). Consider all the n_x edges adjacent to x. Since G has maximum degree 3, we have that at least $n_x - 4$ (one edge connects x to y) of these edges incident to x are edges whose capacity is saturated by saturating flows. The remaining edges different from (x, y) (at most 3) have congestion at least $\frac{4}{5}$. If x splits $((x, y), \cdot)$ among 5 of its neighbors, then it is sending a flow with size greater than 0 through an edge that already has congestion 1. This is a contradiction, since we assumed that $mc^*((H', D')) \leq 1$. Otherwise, x sends at least a fraction $\frac{1}{4}$ of flow $((x, y), \frac{1}{2})$ through at least an edge that already has congestion at least $\frac{4}{5}$, which is a contradiction since we assumed that $mc^*((H', D')) \leq 1$.

Hence, for each saturating flow $((x, y), \frac{1}{2})$, where $(x, y) \in E(H')$, we have that A_{xy} consists of a single edge (x, y). Consider now $A = A_{\phi'(\phi(s))\phi'(\phi(t))}$. Observe that A contains an edge (x, y) only if there exists an edge $e \in E(G)$ such that there exists an edge e' of H traversed by $\phi((x, y))$ and (x, y) is traversed by $\phi'(e')$. We now compute an (s, t)-DAG A^* of (I, f) such that $OPT_{MC}(I) \leq 1$, which is a contradiction. For each edge $e \in E(G)$ such that for every $e' \in E(G')$ traversed by path $\phi(e)$, $\phi'(e')$ is contained in A, add e into $E(A^*)$. Observe that, since ϕ and ϕ' defines a subdivion of G, we have that if a fraction q of the flow from $\phi'(\phi(s))$ to $\phi'(\phi(t))$ is routed through $\phi'(e')$, then the same fraction q of the flow from s to t is routed through e, where $\phi(e)$ traverses e'. Hence, since congestion of $\phi'(e')$ is at most 1, it implies that

$$2 \cdot \frac{2}{5} \left(\frac{c_{max} - c_G((u, v))}{4c_{max}} + 1 \right) + q \frac{1}{5c_{max}} \leq 1$$

$$1 - \frac{c_G((u, v)) - q}{c_{max}} \leq 1$$

$$c_G((u, v)) \geq q,$$

which means that each edge has congestion less than 1, i.e., $OPT_{MC}(I) \leq 1$, a contradiction. Hence, the statement of the theorem holds. ■

The theorem easily follows by Lemma 7.3 and Lemma 7.4. Observe also that since the embedded instance I has degree at most 3, hypercube H' has dimension polynomial w.r.t. the size of I. ■

Appendix H
Min-Congestion-Unsplittable-Flow 2-inapproximability

Refer to Appendix A for notation and terminology details. We now prove that even in a simple 2-FCN, i.e., a complete bipartite graph, it is NP-Hard to approximate MCUF within a factor of 2. We prove it by a polynomial time reduction from the 3-Edge-Coloring problem. The input of 3-Edge-Coloring consists of an unoriented graph G and a set of 3 colors $\{c_1, c_2, c_3\}$. Each edge can be colored with any of these colors. An *edge-coloring* of G assigns a color to each edge of G. An edge-coloring is *valid* if, for each vertex $v \in V(G)$, no pair of edges incident to v have the same colour. If there exists a valid edge-coloring of G, G is 3-*colorable*. In 3-Edge-Coloring, it is asked to determine whether G is 3-colorable. The following lemma is a well-known result about edge-coloring problems.

Lemma 8.1: [22] It is NP-Hard to determine whether a graph G with maximum degree 3 is 3-colorable.

We now use this result to prove the following theorem.

Theorem 8.2: It is NP-Hard to approximate MCUF within a factor of 2, even for a FCN graph F with three vertices in the last stage of F.

Proof: In this proof, each flow demand that we will have to route has size 1. Therefore, we avoid to specify the size of each flow demand and, consequently, the set of flow demands D is modeled simply as a set of pairs of vertices. Let G be an input graph of 3-Edge-Coloring. We construct an instance $I = (F, D)$ of MCUF, where F is a 2-FCN and D is a set of flow demands constructed as follows. Add three vertices y_1, y_2, and y_3 in the last stage of F. For each vertex $v \in V(G)$, add a vertex u_v in the first stage of F. Connect each vertex in the first stage to each vertex in tha last stage, i.e., F is a complete bipartite graph. Each edge of F has capacity 1. We now map each edge of G to a flow demand in D. For each edge $(x, y) \in E(G)$, add a flow demand (u_x, u_y) into D. It is easy to verify that this construction reduction can be done in polynomial time w.r.t. the size of G. Observe that each flow demand must traverse exactly one vertex in the last stage of F in order to be routed between its source and target vertices.

We now prove that G is 3-colorable iff there exists a routing R of flow demands in D such that $mc(I, R) = 1$. Suppose that G is 3-colorable and let γ be a valid edge-coloring of

G that assigns a color $\gamma(e)$ to each edge of G. We construct a routing solution R as follows. We first map colors c_1, c_2, and c_3 to vertices y_1, y_2, and y_3, respectively. Then, for each flow demand (u_x, u_y), if $\gamma((x,y)) = c_i$, with $i = 1,2,3$, we route (u_x, u_y) through y_i. Suppose, by contradiction, that there exists an edge of F that has congestion 2. This implies that at least two flow demands d_1 and d_2 that share at least one endpoint v, are routed through the same vertex y_i, with $i = 1,2,3$. By construction of R, this implies that d_1 and d_2 are both incident to v in G and γ colors both d_1 and d_2 with the same color, which is a contradiction since γ is a valid edge-coloring. Suppose now that all flow demands in D can be routed with congestion at most 1. We construct a edge-coloring γ of G as follows. For each flow demand (u_x, u_y), let y_i be the vertex in the last stage of F, with $i = 1,2,3$, that is traversed by (u_x, u_y). Let $\gamma((x,y)) = c_i$. Suppose, by contradiction, that γ is not a valid edge-coloring of G. Let (x,y) and (x,z) be two edges of G such that $\gamma((x,y)) = \gamma((x,z))$. By construction of γ, flow demands (u_x, u_y) and (u_x, u_z) are both routed to the same vertex y_i, with $i = 1,2,3$. This implies that there are two flows routed through edge (u_x, y_i), which is a contradiction.

The above reduction shows that, if G is 3-colorable, F has congestion at most 1, otherwise, if G is not 3-colorable, F has congestion at least 2 since at least two flows are routed through the same edge. Hence, by Lemma 8.1, the hardness of the problem is proved. ■

APPENDIX I
TIGHTNESS ANALYSIS OF EQUILIBRIUM-ALGO

Refer to Appendix A for notation and terminology details.

EQUILIBRIUM-ALGO analysis is tight in the case of equal size flows. We construct an instance $I = (F, D)$ of MCUF and a routing solution R of I such that there exists a routing solution R such that $mc(I, R) = 1$ and there exists a routing solution R' such that R' is an equilibrium and $mc(I, R') = 4$. This proves that the result in Corollary 5.4 is tight.

We construct $I = (F, D)$ based on the following recursive construction. Let $k = 4n$, for a certain $n > 0$. $I_0 = (F_0, \varnothing)$ consists of a $(k,k,k,1)$-FCN and an empty set of flow demands. $I_n = (F_n, D_n)$ is a $(k,k,k,1+k^2k'_r)$-FCN, where k'_r is the number of $(2,k,k)$-FCN of I_{n-1}, constructed as follow. We denote the first $(2,k,k)$-FCN of F_n by F^1 and the following $k^2k'_r$ $(2,k,k)$-FCN by $F_{i,j}$, with $i = 1,\ldots,k$ and $j = 1,\ldots,k$. The main idea is to map flow demands of several I^-_{n-1} instances to distinct set of k'_r consecutive $(2,k,k)$-FCN of F_n. Let $\phi(x,y) = ((x-1)k + y - 1)k'_r + 1$. For each $i = 1,\ldots,k$ and $j = 1,\ldots,k$, denote by $N_{i,j}$ the subgraph of F_n induced[1] by vertices in $V(F_{\phi(i,j)}) \cup \ldots \cup V(F_{\phi(i,j)+k'_r-1})$ and vertices in the last stage of F_n. Intuitively, $\phi(x,y)$ returns the index of the first $(2,k,k)$-FCN of $N_{x,y}$. Let D_{n-1} be the set of flow demand of a I_{n-1} instance. We now map flows in D_{n-1} to sets of k'_r consecutive $(2,k,k)$-FCN of F_n. For each $i = 1,\ldots,k$ and $j = 1,\ldots,k$, for each flow demand $((b^u_h, b^y_g), f) \in D_{n-1}$, add $((b^{u+\phi(i,j)}_h, b^{y+\phi(i,j)}_g), f)$ into D_n. Observe that, by construction of D_n, there does not exist a flow demand in D_n that has b^1_i as an source vertex, for any $i = 1,\ldots,k$. We creates flow demands from these vertices as follows. For each $1 \le i \le k$, for each $j = 1,\ldots,k$, if $i \neq 1 \wedge j \neq k$, add a flow demand d from vertex b^1_1 of F_n to vertex $b^{\phi(i,j)}_1$ into D_n. Hence, $\phi(i,j)$ returns the index of the first $(2,k,k)$-FCN of F_n that contains the target vertex of the j-th flow demand that has b^1_i as source vertex. We denote by $\rho(d)$ the subgraph $N_{i,j}$. Let $\bar{D}_n \subseteq D_n$ be the set of flow demands that have b^i_1 as a source vertex, with $i = 1,\ldots,k$. Observe that, for each $i = 1,\ldots,k$ and $j = 1,\ldots,k$, there exists at most one flow demand in $\bar{D}_n$ that has a target vertex in $N_{i,j}$. This concludes the definition of I_n.

We now prove some properties of I_n, with $n = 1,2,3,4$. We first introduce some terminology. Given a routing solution R of an instance (F, D), we say that a flow demand $d = ((s,t), f_d) \in D$ is *non-reroutable* if EQUILIBRIUM-ALGO cannot reroute it to a less loaded path, i.e., there does not exist a path p between s and t such that $c(p_d) > c(p) + f_d$. A routing solution is in *equilibrium* if every flow in D is non-reroutable. For any $d \in D_n$, we denote by $D(\rho(d))$ the set of flow demands in D_n that have both their source and target vertices in $\rho(d)$ and by $\bar{D}(\rho(d))$ the set of flow demands that have their source vertex in the first $(2,k,k)$-FCN of $\rho(d)$. A *top-down* path of F is a path between a vertex in the last stage of F and a vertex in the first stage of F.

We first prove that, for any $n \ge 0$, I_n admits a solution with congestion exactly 1.

Lemma 9.1: Given an I_n instance and a top-down path p of I_n to b^1_1, there exists a routing solution of I_n with congestion 1 and $c(p) = 0$.

Proof: We prove that, given an I_n instance, with $n \ge 0$, given a top-down path p of I_n, there exists a routing solution R of I_n such that $mc(I_n, R) = 1$ and $c(p) = 0$. We prove it by induction on n. If $n = 0$, the statement trivially holds since there is no flow routed through I_0. If $n \ge 1$, by inductive hypothesis, we have that for each subgraph $N_{i,j} = (F_{i,j}, D_{i,j})$, with $i = 1,\ldots,k$ and $j = 1,\ldots,k$, contained in I_n, there exists a routing solution R_{n-1} of $N_{i,j}$ such that, given an arbitrary top-down path p' of flows in $D(N_{i,j})$, each edge of $N_{i,j}$ has congestion at most 1 and $c(p') = 0$. We use this inductive hypothesis in order to build a routing solution for I_n with congestion 1 and $c(p) = 0$. Hence, we now show how to route flows in $\bar{D}_n$ and then we use the inductive hypothesis for routing flows in $D(N_{i,j})$, with $i = 1,\ldots,k$ and $j = 1,\ldots,k$. Let $p = (t^g_l, m^1_g, b^1_1)$ be the given top-down path, with $g = 1,\ldots,k$ and $l = 1,\ldots,k$. For each $i = 1,\ldots,k$, consider flow demands $d^i_1,\ldots,d^i_l$ in $\bar{D}_n$, with $l \in \{k-1, k\}$. If $i = 1 \wedge g \neq k$, route d^1_g through (b^1_1, m^1_k, t^k_1) and through an arbitrary top-down path $p_{1,g}$ from t^k_1 to its destination $b^{\phi(1,g)}_1$. If $i > 1 \wedge$, for each $j = 1,\ldots,k$, if $i = l$ route d^l_g through (b^1_i, m^1_g, t^g_1) and through an arbitrary

[1]Given a graph G and a set of vertices $V' \subseteq V(G)$, a subgraph G' of G *induced* by V' is defined as $V(G') = V'$ and $E(G') = \{(x,y) \in E(G) | x \in V' \wedge y \in V'\}$

top-down path $p_{l,g} \neq p$ from t_1^g to its destination $b_1^{\phi(l,g)}$, otherwise, if $i \neq l$, route d_j^i through (b_i^1, m_j^1, t_i^j) and through an arbitrary top-down path $p_{i,j} \neq p$ from t_i^j to its destination $b_1^{\phi(i,j)}$ Observe that, by inductive hypothesis, there exists a routing solution of flow demands in $D(N_{i,j})$ such that it has congestion at most 1 and $c(p_{i,j}) = 0$, with $i = 1, \ldots, k$ and $j = 1, \ldots, k$. Hence, since each flow in $\bar{D}_n$ is routed through a distinct path, we have that the congestion of I_n is at most 1 and $c(p) = 0$. ■

We now prove that there exists a routing solution R of I_4 such that $mc(I_4, R) = 4$ and R is an equilibrium. We first introduce the following three lemmas and then we use them to prove construct such instance I_4.

Lemma 9.2: Given a instance $I_n = (F_n, D, n)$, consider a routing R' where only flow demands in $\bar{D}_n$ are routed. For any $d = (s,t) \in \bar{D}_n$ such that $c(p_d) \leq n$ and $c(p_d) \leq 2$, there exists a routing solution R of I_n such that d and every flow demand in $D(\rho(d))$ is non-reroutable and each flow in D_n is routed according to R'.

Proof: We prove it by induction on n. In the base case $n = 0$, it trivially holds since D_0 of I_0 is the empty set. In the inductive step $n > 0$, consider an arbitrary routing where only flows in $\bar{D}_n$ are routed through F_n. Consider a flow demand $d \in \bar{D}_n$. If $c(p_d) = 1$, then it is easy to see that it is not possible to reroute d in order to obtain a better routing solution. In fact, for every $d \in \bar{D}_n$, we always have that $c(p_d) \geq 1$. Otherwise, if $c(p_d) = 2$, suppose that d is routed through a path (b_1^g, m_l^g, t_h^l), with $1 \leq g \leq k$, $1 \leq l \leq k$, and $1 \leq h \leq k$. Let b_1^i be a vertex of the first $(2,k,k)$-FCN of $\rho(d)$. Observe that d is routed through $p' = (t_h^l, m_l^g, b_1^g)$ in $\rho(d)$. We consider an arbitrary routing R' of flow demands in $\bar{D}(\rho(d))$ such that (i) only d is routed through p', (ii) b_1^i routes its $k-1$ flow demands $d_1, \ldots, d_{k-1}$ through its $k-1$ neighbors $m_1^1, \ldots, m_{g-1}^1, m_{g+1}^1, \ldots, m_k^1$, and (iii) the value of the most congested edge in $\rho(d)$ is less than 2. This implies that, by construction of R, d is non-reroutable and, by inductive hypothesis, for every $d' \in \bar{D}(\rho(d))$, d' and every flow demand in $D(\rho(d'))$ is non-reroutable. Hence, every flow demand in $D(\rho(d))$ is non-reroutable, which proves the statement of the lemma also in this case. ■

Lemma 9.3: Given a I_n instance, with $n \geq 3$, consider a routing R' where only flow demands in $\bar{D}_n$ are routed such that: (i) for each $i = \frac{k}{2}+1, \ldots, k$, edges $(m_i^1, t_1^i), \ldots, (m_i^1, t_{\frac{k}{2}}^i)$ have congestion 3, and (ii) edge $(m_{\frac{k}{2}}^1, t_1^{\frac{k}{2}})$ ha congestion 2. Consider a flow demand $d = (s,t) \in \bar{D}_n$, with $c(p_d) = 3$, such that d is routed neither through $m_{\frac{k}{2}}^1$ nor through t_y^x, for any $x = \frac{k}{2}+1, \ldots, k$ and $y = \frac{k}{2}+1, \ldots, k$. There exists a routing solution R of I_n such that d and every flow demand in $D(\rho(d))$ are non-reroutable and each flow in D_n is routed according to R'.

Proof: We compute such routing solution R as follows. Assume that d is routed through a path (b_1^g, m_l^g, t_h^l), where $m_l^g \neq m_{\frac{k}{2}}^1$ and $t_h^l \neq t_y^x$, and this path has congestion less than 4. Let b_1^i be a vertex of the first $(2,k,k)$-FCN of $\rho(d)$. Observe that d is routed through $p' = (t_h^l, m_l^g, b_1^g)$ in $\rho(d)$. For each $j = \frac{k}{2}+1, \ldots, k$, let $d_1^j, \ldots, d_k^j$ be flow demands in $\bar{D}(\rho(d))$ that have b_j^i as source vertex. For each $l = 1, \ldots, k$, if $\lceil \frac{l}{2} \rceil + \frac{k}{2} \neq j$, let $\bar{l} = \lceil \frac{l}{2} \rceil + \frac{k}{2}$ and route d_l^j through $(m_{\bar{l}}^j, t_j^{\bar{l}})$. Otherwise, if $\lceil \frac{l}{2} \rceil + \frac{k}{2} = j$, route d_l^j through $(m_{\frac{k}{2}}^j, t_j^{\frac{k}{2}})$. For each $j = 2, 3$, let $d_1^j, \ldots, d_k^j$ be flow demands in $\bar{D}(\rho(d))$ that have b_j^i as source vertex. For each $l = \frac{k}{2}+1, \ldots, k$, route d_l^j through (m_l^j, t_l^l). For each $j = 2, \ldots, \frac{k}{2}$, let $d_1^j, \ldots, d_k^j$ be flow demands in $\bar{D}(\rho(d))$ that have b_j^i as source vertex. Route d_1^j and d_2^j through $(m_{\frac{k}{2}}^j, t_j^{\frac{k}{2}})$. Let $d_1^1, \ldots, d_k^1$ be flow demands in $\bar{D}(\rho(d))$ that have b_1^i as source vertex. For each $l = 1, \ldots, k-1$, let $\bar{l} = \lceil l \frac{4}{k} \rceil$ and route d_l^1 through $(m_{\bar{l}}^1, t_1^{\bar{l}})$. Observe that $c(p') = 0$. Now, route all the remaining flows in $\bar{D}(\rho(d))$ that were not routed so far in such a way that the congestion of edges in $\rho(d)$ is at most 2 and $c(p') = 0$. This concludes the definition of R. By construction of R, d is non-reroutable and, by Lemma 9.2, for every $d' \in \bar{D}(\rho(d))$ which has $c(p') \leq 2$, there exists a routing of flows in $D(\rho(d'))$ such that d' and every flow demand in $D(\rho(d'))$ is non-reroutable and every flow is routed according to R. Hence, every flow demand in $D(\rho(d))$ is non-reroutable, which proves the statement of the lemma also in this case. ■

It is easy to see that, by a symmetry argument, Lemma 9.3 can be used to prove the following lemma.

Lemma 9.4: Given a I_n instance, with $n \geq 3$, consider a routing R' where only flow demands in $\bar{D}_n$ are routed such that: (i) for each $i = \frac{k}{2}+1, \ldots, k$, either edges $(m_i^1, t_{\frac{k}{2}+1}^i), \ldots, (m_i^1, t_k^i)$ have congestion 3, and (ii) edge $(m_{\frac{k}{2}}^1, t_1^{\frac{k}{2}})$ ha congestion 2. Consider a flow demand $d = (s,t) \in \bar{D}_n$, with $c(p_d) = 3$, such that d is routed neither through $m_{\frac{k}{2}}^1$ nor through t_y^x, for any $x = \frac{k}{2}+1, \ldots, k$ and $y = 1, \ldots, \frac{k}{2}$. There exists a routing solution R of I_n such that d and every flow demand in $D(\rho(d))$ are non-reroutable and each flow in D_n is routed according to R'.

We can now exploit Lemma 9.2, Lemma 9.3, and Lemma 9.4 in order to build a routing solution R of I_4 such that at least an edge has congestion 4 and R is in an equilibrium.

Lemma 9.5: There exists a routing solution of I_4 that has congestion 4.

Proof: We compute such routing solution R as follows. Let $d_1^1, \ldots, d_k^1$ be flow demands in $\bar{D}_4$ that have b_1^1 as source vertex. For each $j = 1, \ldots, 3\frac{k}{4}$, let $\bar{j} = \lceil j \frac{3}{k} \rceil$ and route d_j^1 through $(m_{\bar{j}}^1, t_1^{\bar{j}})$. We have that edges $(b_1^1, m_{\frac{k}{4}+1}^1), \ldots, (b_1^1, m_{\frac{k}{2}}^1)$ have congestion 3. Route $d_{3\frac{k}{4}+1}^1$ though (m_1^1, t_1^1). Both edges (b_1^1, m_1^1) and (m_1^1, t_1^1) have congestion 4. For each $i = \frac{k}{4}+1, \ldots, k$, let $d_1^i, \ldots, d_k^i$ be flow demands in $\bar{D}_4$ that have b_i^1 as source vertex. For each $j = 1, \ldots, k$, let $\bar{j} = \left((j-1) mod \frac{k}{2}\right) + \frac{k}{2} + 1$, $\bar{i} = 2\left(i \ mod \ \frac{k}{4}\right) + \lceil j \frac{2}{k} \rceil$ and route d_j^i through $(b_i^1, m_{\bar{j}}^1, t_{\bar{i}}^{\bar{j}})$. We have that, for each $j = \frac{k}{2}+1, \ldots, k$, edges $(m_j^1, t_1^j), \ldots, (m_j^1, t_{\frac{k}{2}}^j)$ have congestion 3. Let d_1^2 and d_2^2 be two flow demands in $\bar{D}_4$

that have b_2^1 as source vertex. Route d_1^2 and d_2^2 through $(m_{\frac{k}{2}}^1, t_1^{\frac{k}{2}})$. Route all the remining flows in $\bar{D}_4$ through any path that does not traverse any vertex m_j^1, with $j \geq \frac{k}{2}$, in such a way that the congestion created by these flow demands is less than 4. Observe that, edges $(b_1^1, m_1^1), \ldots, (b_1^1, m_{\frac{k}{4}}^1)$ have have congestion 4. Consider any flow demand d in $\bar{D}_4$ that is routed through any of these edges. Observe that d is routed in $\rho(d)$, through $(t_1^1, m_1^{i^*}, b_1^{i^*})$, where i^* is the index of a vertex $b_1^{i^*}$ of the first $(2, k, k)$-FCN of $\rho(d)$ in I_4. We construct a routing solution R' of flows in $D(\rho(d))$ that is similar to R. Let $d_1^{i^*}, \ldots, d_k^{*}$ be flow demands in $\bar{D}(\rho(d))$ that have $b_1^{i^*}$ as source vertex. For each $j = 1, \ldots, 3\frac{k}{4}$, let $\bar{j} = \lceil j\frac{3}{k} \rceil + \frac{k}{4}$ and route $d_j^{i^*}$ through $(m_{\bar{j}}^{i^*}, t_1^{\bar{j}})$. We have that edges $(b_1^{i^*}, m_{\frac{k}{4}+1}^{i^*}), \ldots, (b_1^{i^*}, m_{\frac{k}{2}}^{i^*})$ have congestion 3. For each $i = \frac{k}{4} + 1, \ldots, k$, let $d_1^i, \ldots, d_k^i$ be the flow demands in $\bar{D}_4$ that have b_i^1 as source vertex. For each d_j^i, with $j = 1, \ldots, k$, and t_y^x be a vertex traversed by $p_{d_j^i}$. Let $d_1^{i^*+i}, \ldots, d_k^{i^*+i}$ be the flow demands in $\bar{D}(\rho(d))$ that have $b_i^{i^*}$ as source vertex. Route $d_l^{i^*}$, with $l = 1, \ldots, k$, through t_{k-y+1}^x. We have that, for each $j = \frac{k}{2} + 1, \ldots, k$, edges $(m_j^{i^*}, t_{\frac{k}{2}+1}^{i^*}), \ldots, (m_j^{i^*}, t_k^j)$ have congestion 3. Let $d_1^{i^*}$ and $d_2^{i^*}$ be two flow demands in $\bar{D}(\rho(d))$ that have $b_2^{i^*}$ as source vertex. Route $d_1^{i^*}$ and $d_2^{i^*}$ through $(b_2^{i^*}, m_{\frac{k}{2}}^{i^*}, t_1^{\frac{k}{2}})$, exactly as we have done in $\bar{D}_4$. We have that, edges $(m_{\frac{k}{2}}^{i^*}, t_2^{\frac{k}{2}}), \ldots, (m_{\frac{k}{2}}^{i^*}, t_k^{\frac{k}{2}})$ have congestion 2. Route all other flows in $\bar{D}(\rho(d))$ through any path that does not traverse any vertex $m_j^{i^*}$, with $j \geq \frac{k}{2}$ in such a way that the congestion created by these flow demands is less than 4. Observe that, by construction of R and R', d is non-reroutable and, by Lemma 9.4, there exists a routing solution of flows in $D(\rho(d))$ such that every flow demand in $D(\rho(d))$ is non-reroutable and every flow demand in $D(\rho(d))$ is routed according to R'. Moreover, for every flow demand $d'\bar{D}_4$ that is routed through a path with congestion 3, by Lemma 9.3, there exists a routing R'' of flows in $D(\rho(d'))$ such that d' and every flow demand in $D(\rho(d''))$ are non-reroutable and every flow demand in $D(\rho(d))$ is routed according to R''. For all the remaing flow demands $d'' \in \bar{D}_4$ that are routed through a path that has congestion 2 or 1, by Lemma 9.2, there exists a routing solution R''' of flows in $D(\rho(d''))$ such that d'' and every flow demand in $D(\rho(d''))$ are non-reroutable and every flow demand in $D(\rho(d''))$ is routed according to R'''. Hence, R is in an equilibrium and the lemma is proved. ■

The following theorem is a direct consequence of Lemma 9.1 and Lemma 9.5.

Theorem 9.6: There exists an instance I of MCUF such that $mc^*(I) = 1$ and EQUILIBRIUM-ALGO returns a routing solution R such that $mc(I, R) = 4$.