Improved Throughput for All-or-Nothing Multicommodity Flows With Arbitrary Demands

Anya Chaturvedi, Chandra Chekuri, Mengxue Liu, Andréa W. Richa, Matthias Rost, Stefan Schmid, and Jamison Weber

***Abstract*— Throughput is a main performance objective in communication networks. This paper considers a fundamental maximum throughput routing problem — the *All-or-Nothing Multicommodity Flow (ANF)* problem — in arbitrary *directed graphs* and in the practically relevant but challenging setting where demands can be (much) larger than the edge capacities, mandating the need for *splittable* flows (i.e., flows may not follow a single path). Formally, the input for the ANF problem is an edge-capacitated directed graph where we have a given number of source-destination node-pairs with their respective demands and strictly positive weights. The goal is to route a maximum weight subset of the given pairs (i.e., the weighted *throughput*), respecting the edge capacities: A commodity is routed if all of its demand is routed from its respective source to destination (this is the all-or-nothing aspect). We present a polynomial-time bi-criteria approximation randomized rounding framework for this NP-hard problem that yields an *arbitrarily good approximation on the weighted throughput while violating the edge capacity constraints by at most a sublogarithmic multiplicative factor*. We present two non-trivial linear programming relaxations that can be used in the framework; the first uses a novel edge-flow formulation and the second uses a packing formulation. We demonstrate the "equivalence" of these formulations and then highlight the advantages of each of the two approaches. We complement our theoretical results with a proof of concept empirical evaluation, considering a variety of network scenarios.**

***Index Terms*— Multicommodity flows, randomized and approximation algorithms, network optimization.**

I. Introduction

THE study of routing and multicommodity flow problems is motivated by many real-world applications, such as in the optimization of communication and traffic networks, as well as by the crucial role flows and cuts play in combinatorial optimization [11]. In this paper, we are interested in throughput optimization in the context of communication networks serving multiple commodities. Throughput is a most fundamental performance metric in many networks [27], and we are particularly interested in the practically relevant scenario where flows have certain minimal performance or quality-of-service requirements, in the sense that they need to be served in an *all-or-nothing* manner with respect to their demands.

Our problem belongs to the family of all-or-nothing (splittable) multicommodity flow problems. In contrast to most existing literature, we consider a more realistic model in the following respects:

- The underlying communication graph can be *directed*. This is motivated by the fact that in most practical communication networks (e.g., optical networks or wireless networks), the available capacities in the different link directions can differ.
- A single commodity demand can be larger than the capacity of any single link or path. Consider for example a bulk transfer, or the fact that traffic patterns are often highly skewed, with a small number of elephant flows consuming a significant amount of bandwidth resources [34]. Only *splittable* flows can serve such demands.
- The total demand can be larger than the network capacity. To make efficient use of the given network resources, we hence need a clever *admission control* mechanism, in addition to a routing algorithm.

We define the *All-or-Nothing (Splittable) Multicommodity Flow (ANF)* problem as follows: It takes as input a flow network modeled as a capacitated directed graph $G(V, E)$, where V is the set of nodes, E is the set of edges, and each edge e has a capacity $c_e > 0$; we are also given a set of source-destination pairs (s_i, t_i), where $s_i, t_i \in V$, $i \in [k]$,[1] each with a demand $d_i > 0$ and weight $w_i > 0$. Let $n = |V|$ and $m = |E|$. The edge capacities c_e, the demands d_i and the weights w_i can be arbitrary positive functions on n and k, for any $e \in E$ and $i \in [k]$. A valid set of flows for commodities $1, \ldots, k$ in G (i.e., a valid *multicommodity flow*), must satisfy standard flow conservation constraints for each commodity i—which imply that the amount of flow for commodity i entering a node v has to be equal to the flow for commodity i leaving v, if $v \neq s_i, t_i$—and also that the *load* of any edge e, given by the sum of the flows for all commodities on e, must not exceed the edge's capacity c_e. Commodity i is *satisfied* if d_i units of flow of this commodity are successfully routed from s_i to t_i in the network. These constraints are reflected in our mixed integer program edge-flow formulation in Figure 1.

[1]Let $[x]$ denote the set $\{1, \ldots, x\}$, for any positive integer x.

Manuscript received 29 November 2022; revised 14 July 2023; accepted 15 September 2023; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor R. Pedarsani. Date of publication 7 November 2023; date of current version 18 April 2024. This work was supported in part by NSF under Grant CCF-1637393, Grant CCF-1733680, and Grant CCF-1910149; and in part by DoD-Army Research Office (ARO) Multidisciplinary University Research Initiative (MURI) under Award W911NF-19-1-0233. An earlier version of this paper was presented in part at the IEEE INFOCOM 2019-IEEE Conference on Computer Communications and ACM SIGMETRICS Performance Evaluation Review 49 (3) [DOI: 10.1145/3529113.3529121]. *(Corresponding author: Jamison Weber.)*

Anya Chaturvedi, Mengxue Liu, Andréa W. Richa, and Jamison Weber are with the School of Computing and Augmented Intelligence, Arizona State University, Tempe, AZ 85281 USA (e-mail: anya.chaturvedi@asu.edu; mengxue.liu@asu.edu; aricha@asu.edu; jwweber@asu.edu).

Chandra Chekuri is with the Department of Computer Science, University of Illinois at Urbana-Champaign, Champaign, IL 61820 USA (e-mail: chekuri@illinois.edu).

Matthias Rost is with Observe Inc., San Mateo, CA 94402 USA (e-mail: mrost@inet.tu-berlin.de).

Stefan Schmid is with the Faculty IV, Department of Telecommunication Systems, Technische Universität Berlin, 10623 Berlin, Germany (e-mail: schmiste@gmail.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNET.2023.3325437, provided by the authors.

Digital Object Identifier 10.1109/TNET.2023.3325437

$$\max \sum_{i=1}^{k} w_i f_i \tag{1}$$

$$\sum_{(s_i,v)\in E} f_{i,(s_i,v)} = f_i \qquad 1 \le i \le k \tag{2}$$

$$\sum_{(u,v)\in E} f_{i,(u,v)} = \sum_{(v,u)\in E} f_{i,(v,u)} \quad 1 \le i \le k, \forall v \in V \setminus \{s_i, t_i\} \tag{3}$$

$$d_i \sum_{i=1}^{k} f_{i,(u,v)} \le c_{(u,v)} \qquad (u,v) \in E \tag{4}$$

$$d_i \cdot f_{i,(u,v)} \le c_{(u,v)} \cdot f_i \qquad 1 \le i \le k, (u,v) \in E \tag{5}$$

$$f_{i,(u,v)} \ge 0 \qquad 1 \le i \le k, (u,v) \in E \tag{6}$$

$$f_i \in \{0,1\} \qquad 1 \le i \le k \tag{7}$$

Fig. 1. Compact Edge-Flow ANF Formulation; its LP relaxation is obtained simply by relaxing the variables f_i to be drawn from the unit interval $[0, 1]$.

We aim to maximize the total profit of a subset of commodities that can be concurrently satisfied in a valid multicommodity flow. Specifically, the goal is to find a subset $K' \subseteq [k]$ of commodities to be concurrently satisfied such that the (weighted) *throughput*, given by $\sum_{i\in K'} w_i$, is maximized over all possible K'. The flow can be *split* arbitrarily along many branching routes (subject to flow conservation and edge capacity constraints) and needs not to be necessarily integral.

The ANF problem was introduced in [14] as a relaxation of the classical Maximum Edge-Disjoint Paths problem (MEDP) and is known to be NP-Hard and APX-hard even in the restricted setting of unit demands and when the underlying graph is a tree [14], [21]. In directed graphs, the problem is hard to approximate to within an $n^{\Omega(1/c)}$ factor, even when restricted to unit demands and when edge capacities are allowed to be violated by a factor c [15]. When demands can exceed the minimum capacity, strong lower bounds exist in even more restricted settings [35].

Given the hardness of approximating the ANF problem the literature has followed a bi-criteria optimization approach where edge capacities can be violated slightly. Moreover, augmentation, in the context of network resource availability, is often relevant in practice, especially in virtualized and cloud environments. Distributed systems are increasingly virtualized, and users typically rent or buy a certain share of the physical resources. Accordingly, extra resources can be bought when needed [2], [20], [36]. In addition, we can view augmentation analysis as some kind of sensitivity analysis: if with a little augmentation of the edge capacities, we are as good as an optimal algorithm, it means that the algorithm is robust and intuitively provides good approximations.

In this paper, we seek an (α, β)-*approximation algorithm*: For parameters $\alpha \in (0, 1]$ and $\beta \ge 1$, we seek a *polynomial-time* algorithm that outputs a solution to the ANF problem whose throughput is at least an α *fraction of the maximum throughput* and whose *load on any edge* e *is at most* β *times the edge capacity* c_e, with high probability.[2] The parameter β hence provides an upper bound on the *edge capacity violation ratio* (or *congestion*) incurred by the algorithm.

A. Our Contributions

This paper revisits a fundamental maximum throughput routing problem, the all-or-nothing multicommodity flow (ANF) problem, considering a more general and practical setting where the network topology can be an *arbitrary directed graph*, with *arbitrary, non-uniform commodity demands* that can be much larger than the edge capacities, in contrast to most of the existing work in the literature. This model is challenging as it not only requires a clever algorithm to efficiently route the splittable commodities across the directed and capacitated network, but also an admission control policy.

We make several contributions. On the theoretical side, we present a bi-criteria approximation randomized rounding framework for this NP-hard problem that achieves an arbitrarily good approximation of the throughput while only violating the edge capacities by a sublogarithmic factor. More specifically,

- We present *two non-trivial ANF linear programming relaxations*: One is a *strengthened relaxation* of a *compact edge-flow* mixed integer program (MIP) formulation that allows for easy solving via standard LP solvers, while the other is an exponential-size formulation (solvable in polynomial time using a separation oracle) that considers a *"packing"* view and allows for simpler proofs and a more flexible approach. We prove the non-trivial *equivalence* of the two relaxations and highlight the advantages of each of the two approaches.
- Via these relaxations, we obtain a polynomial-time *randomized rounding* algorithm that yields an $(1 - \epsilon)$ *throughput approximation*, for any $1/m \le \epsilon < 1$, with an *edge capacity violation ratio* (or *congestion*) of $O(\min\{k, \log n/\log\log n\})$, with high probability.
- We also present a *deterministic rounding algorithm by derandomization*, using the method of pessimistic estimators. Contrary to most algorithms obtained this way, our derandomized algorithm is simple enough to be also of relevance in practice.

In addition, our packing framework for ANF has interesting networking applications, beyond the specific model considered in this paper. We discuss different examples, related to *unsplittable flows*, flows that are *split into a small number of paths*, *routing along disjoint paths* for fault-tolerance, using *few edges for the flow*, or routing flow along *short paths*.

As a proof of concept, we show how to *engineer our algorithms* for practical scenarios. To this end, we couple three algorithms that allow one to compute the relaxed LP solutions efficiently, in terms of time and space, with both our randomized and derandomized algorithms. The first algorithm *directly solves the compact ANF formulation* using an off-the-shelf solver, in our case CPLEX; the second algorithm approximately solves the packing LP relaxation via a well-known *multiplicative weight update (MWU)* approach, based on Lagrangean relaxation; the last and third algorithm is a faster MWU-based heuristic called *permutation routing*. We provide general guidelines about the relative efficacy of these algorithms in specific real-world networks. As a contribution to the research community, to ensure reproducibility and facilitate follow-up work, our implementation (source code) and experimental artifacts is available at [9].

B. Novelty and Related Work

We presented preliminary results leading to this journal article at two conferences, IEEE INFOCOM 2019 [26] and PERFORMANCE 2021 [10]. In particular, our compact edge-flow LP formulation first appeared in [26] and led to $(1/3, O(\sqrt{k \log n}))$-approximation guarantees for the ANF problem for the case of *uniform* demands and weights in directed graphs (note that while [26] considered the case of uniform demands, there was also no restriction on how

[2]With probability at least $1 - 1/n^c$, where $c > 0$ is a constant.

large these demands could be). In this article, we significantly improve and generalize the randomized rounding framework outlined in [26], in several ways: (a) We are able to achieve an arbitrarily good throughput approximation bound; (b) our bound on the edge capacity violation ratio does not depend on the number of commodities k,[3] and significantly improves on the bound of $O(\sqrt{k \log n})$in [26]; and (c) we were able to accommodate arbitrary non-uniform demands and commodity weights. In addition, we provide a derandomized algorithm for the ANF problem and a more flexible packing MIP formulation for the ANF problem that leads to several interesting extensions of practical interest. Some of these ideas were sketched in our previous short paper [10], however, without any technical details, proofs or evaluations.

Other work on bicriteria (α, β)-approximation schemes for the ANF problem that are closely related to ours aim at keeping β constant, while letting α be a function of n. The work of Chekuri et al. [13], [14] is the most relevant and was also the first to formalize the ANF problem. Their work implies an approximation algorithm for the general (weighted, non-uniform demands) ANF problem in *undirected graphs* with $\alpha = \Omega(1/\log^3 k)$ and $\beta = 1$. A requirement of their algorithm is that $\max_i d_i \leq \min_e c_e$. This is a strong assumption, since it eliminates all (undirected) networks G where the above assumption fails, such as for example complete graphs with unit edge capacities and demands $2 \leq d_i \leq n-1$, for all i. Hence, besides the fact that our approximation guarantees differ from those of [14] (we have an arbitrarily good throughput approximation for any $0 < \alpha < 1 - 1/m$ and logarithmic β, while they achieve constant β at the expense of a polylogarithmic $1/\alpha$), our results also apply to *any directed graph* G, without any assumptions on how d_i compares to individual edge capacities. We note that even in undirected graphs and unit demands, the ANF problem does not admit a constant factor approximation if only constant congestion is allowed [4]. Thus, obtaining a good throughput approximation even in restricted settings requires $\omega(1)$ congestion violation.

The ANF problem gets considerably more challenging in directed graphs. Chuzhoy et al. [15] show that, even if restricted to unit demands, the throughput is hard to approximate to within polynomial factors in directed graphs when constant congestion is allowed. In [11], Chekuri and Ene consider a variation of the ANF problem — the *Symmetric All or Nothing Flow (SymANF)* problem — in *directed* graphs with *symmetric unit demand pairs and unit edge capacities*, also aiming at constant β and polylogarithmic $1/\alpha$. In SymANF, the input pairs are unordered and a pair s_i, t_i is routed if and only if both the ordered pairs (s_i, t_i) and (t_i, s_i) are routed; the goal is to find a maximum subset of the given demand pairs that can be routed. The authors provide a poly-logarithmic throughput approximation with constant congestion for SymANF, by extending the well-linked decomposition framework of [13] to the directed graph setting with symmetric demand pairs. However, their approach, like the one for undirected graphs is limited to the setting where $\max_i d_i \leq \min_e c_e$. As explained above, our work considers a more general network setting where demand pairs need not be symmetric and demands values can exceed the capacities. Further, our goal is to obtain an arbitrarily good approximation of the throughput while relaxing the capacity violation which is a different regime.

The *Maximum Edge-Disjoint Paths (MEDP)* [19] problem considers a set of pairs of nodes to be routable if they can be connected using edge-disjoint paths and aims at finding the largest number of routable pairs. The *Unsplittable Flow Problem* (UFP) is a generalization of MEDP to non-uniform demands while requiring that all flow for a pair is routed along a single path. MEDP and UFP are classical routing problems and have been extensively studied in VLSI routing where the constraint of using a single path for connecting pairs is particularly important. MEDP and UFP tend to be harder to approximate than ANF. For instance, even for unit demands and undirected graphs, MEDP is hard to approximate to almost a polynomial factor [16], and in directed graphs the problem is hard to approximate to within an $\Omega(m^{1/2-\epsilon})$-factor [23]. MEDP and UFP have been mostly considered under the no-bottleneck assumption, that is, when $\max_i d_i \leq \min_e c_e$. Without this assumption, UFP becomes hard to approximate to within an $m^{1/2-\epsilon}$ factor even in very restricted settings [35].

Finally, our work leverages randomized rounding techniques presented by Rost et al. [32], [33] in the different context of virtual network embedding problems (i.e., when flow endpoints are subject to optimization).

C. Organization

The remainder of the paper is organized as follows. We introduce our ANF compact edge-flow and packing MIP formulations in Section II. We then present our approximation guarantees based on a randomized rounding approach for the packing MIP in Section III. Section IV presents the randomized rounding algorithm based on the compact LP and its derandomization, while Section V describes the MWU algorithm. We discuss more general applications of our packing framework in Section VI. We report on simulation results in Section VII, and conclude in Section VIII.

II. Mixed Integer Programming Formulations

In this section, we present two non-trivial mixed integer programming (MIP) formulations for the ANF problem, one based on a compact edge-flow formulation and one based on a flow packing formulation. We then show that their linear programming (LP) relaxations are equivalent in the sense that if one formulation has a feasible flow routing with throughput z, so does the other formulation. This allows us to use either formulation, as convenient, to present, prove, and empirically evaluate the approximation guarantees of our randomized rounding approach for the ANF problem.

Recall that the input is a directed graph $G = (V, E)$, with $n = |V|$ and $m = |E|$, edge capacities $c : E \to \mathbb{Z}_+$ and k demand pairs (or *commodities*) (s_i, t_i), $i \in [k]$, each with a non-negative weight w_i and a non-negative integer demand d_i. We say that $f : E \to \mathbb{R}_+$ is a *valid* flow for pair i if f routes d_i units from s_i to t_i in G and respects the edge capacities. Note that if pair i cannot be routed in isolation in G, then we may as well discard it (since there are no valid flows for i; this can be checked in polynomial time for each commodity, via a max-flow algorithm). Thus, w.l.o.g.[4] *we assume that each commodity i admits some valid flow in G.*

[3]Unless k is very small, $o(\log n/\log\log n)$, in which case we get an approximation bound of k.

[4]Without loss of generality.

A. Compact Edge-Flow Formulation

We present our general compact edge-flow based MIP formulation for the ANF problem[5] in Figure 1. We use an indicator variable $f_i \in \{0,1\}$ to indicate whether a commodity i is successfully routed through G. Next, we denote $f_{i,e} \in [0,1]$ as the fraction of flow for commodity i allocated to a particular edge $e \in E$. The total flow assigned to a fixed edge e is given by $\sum_i d_i \cdot f_{i,e}$ and the total weighted throughput is given by $\sum_i w_i f_i$. Constraints (2-4) define the value of the total flow for each commodity i, enforce flow conservation for each i, and stipulate that no edge capacity is violated by the flow assignments. Constraint (5) ensures that for a fixed commodity i, the ratio of flow assigned to an edge e to the total flow of that commodity does not exceed the capacity of e: These constraints are actually redundant for the compact MIP formulation, but will *strengthen the LP relaxation* of the formulation in Figure 1, obtained by allowing each f_i to assume any real value in $[0,1]$. In fact, Constraint (5) is crucial to establishing the equivalence between the LP relaxation of the compact edge-flow and the ANF packing formulations in Section II-C.

The compact edge-flow LP relaxation has size polynomial in n and k and hence can be solved in polynomial time (e.g., using the Ellipsoid method). Moreover, given the compact nature of the LP, one can often use a standard LP solver such as CPLEX in practice.

B. A Packing Framework for ANF

In this section, we describe a *packing* view of the ANF problem that allows for a more flexible approach, as we discuss below and in Section VI. In this formulation, we will be packing an entire flow assignment for each commodity i, selected from the set of all possible valid flows between s_i and t_i. This approach is akin to using the path formulation for flows rather than the edge-based flow formulation. Such a perspective allows one to see why the randomized rounding framework for rounding paths [31] generalizes to rounding flows, and thus allows us to leverage previous work on rounding paths in the literature. We prove the approximation guarantees of our randomized rounding approach in Section III-B.

Let $\mathcal{F}_i$ denote the set of all valid flows for pair i. Each $\mathcal{F}_i$ is not necessarily a finite set. However, we can restrict attention to a finite set of flows by considering the polyhedron of all feasible s_i-t_i flows in G and considering only the finitely many vertices of that polyhedron; any valid flow can be expressed as a convex combination of the flows defined by the polyhedron's vertices. Since the number of vertices (and corresponding flows) of the packing polyhedron can be exponential, this formulation may have exponential size, but we show that its LP relaxation can still be solved in polynomial time via a separation oracle.

We now describe a mixed integer programming formulation that captures the ANF problem. This formulation is very large: In general it can be exponential in n, m and k. For each i, we have a binary variable x_i to indicate whether commodity i is routed or not. For each i and each valid flow $F \in \mathcal{F}_i$, we have a variable y_F to indicate the fraction of x_i that is routed using the flow F. For a flow F we let F_e denote the amount of flow on e used by F; note that F_e is fixed, for each F and e, and hence is not a variable.

[5]The compact MIP formulation presented here generalizes the one in our conference version [26] to accommodate arbitrary demands and commodity weights.

$$(a) \quad \max \sum_{i=1}^{k} w_i x_i \tag{8}$$

$$\sum_{F \in \mathcal{F}_i} y_F = x_i \qquad 1 \le i \le k \tag{9}$$

$$\sum_{i=1}^{k} \sum_{F \in \mathcal{F}_i} F_e \cdot y_F \le c_e \qquad e \in E \tag{10}$$

$$x_i \in \{0,1\} \qquad 1 \le i \le k \tag{11}$$

$$y_F \ge 0 \qquad F \in \mathcal{F}_i, 1 \le i \le k \tag{12}$$

$$(b) \quad \max \sum_{i=1}^{k} w_i \sum_{F \in \mathcal{F}_i} y_F \tag{13}$$

$$\sum_{F \in \mathcal{F}_i} y_F \le 1 \qquad 1 \le i \le k \tag{14}$$

$$\sum_{i=1}^{k} \sum_{F \in \mathcal{F}_i} F_e \cdot y_F \le c_e \qquad e \in E \tag{15}$$

$$y_F \ge 0 \qquad F \in \mathcal{F}_i, 1 \le i \le k \tag{16}$$

Fig. 2. (a) Mixed integer programming formulation for ANF based on "flow" variables; (b) its LP relaxation.

The formulation shown in Figure 2(a) is an exact formulation for the ANF problem. We show this by expanding upon the constraints and the reasoning behind them. We are trying to maximize the weighted subset of commodities being routed where x_i corresponds to the variable denoting whether a commodity is routed or not. Constraint (9) ensures that the sum of fractional flows of a commodity is equal to x_i. Constraint (10) is to ensure the capacity constraints are satisfied for all edges. Constraints (11) and (12) are just to clarify that x_i can only be 0 or 1 since it is an all-or-nothing formulation and that the corresponding y_F's are the fractional flows variables that actually sum up to be the respective x_i. In this formulation, flow conservation constraints are automatically satisfied, since each flow $F \in \mathcal{F}_i$—or when we scale F down uniformly by y_F—is in itself a valid flow for commodity i. Note that the LP relaxation in Figure 2(b), has been simplified, for convenience, by replacing x_i by $\sum_{F \in \mathcal{F}_i} y_F$, thus projecting out the variables x_i.

C. Equivalence of the Two Formulations

Here we prove that the LP relaxation of the packing formulation (Figure 2(b)) is *equivalent* to the LP relaxation of the compact formulation given in Figure 1. By equivalent, we mean the following: Given a feasible solution to one LP we can obtain a feasible solution to the other LP of the same value. One possible way of showing this might be by establishing that the packing formulation can be obtained through a Dantzig-Wolfe decomposition [17], [18] of the compact LP. However, we opted for the simpler proof below. We prove both directions.

First, consider a feasible solution to the compact formulation. For commodity i, let $f_i \in [0,1]$ be the total fraction of d_i that is routed from s_i to t_i, and let $f_{i,e} \in [0,1]$ be fraction of f_i that is assigned to edge $e \in E$, satisfying flow conservation and capacity constraints. We first construct a flow $G_i : E \to \mathbb{R}_+$ of d_i units from s_i to t_i: We set $G_i(e) = d_i f_{i,e}/f_i$, for all $e \in E$. It is easy to verify that G_i is a flow of d_i units from s_i to t_i. Moreover by the strengthening constraint (Constraint ((5)) in Figure 1, we see that $G_i(e) \le c_e$ for all e and hence G_i is a feasible flow in the capacities. Putting together these facts, $G_i \in \mathcal{F}_i$. We obtain a feasible solution to the packing formulation as follows. Let $x_i = \sum_{F \in \mathcal{F}_i} y_F$ be the fraction routed for commodity i in the

packing solution. For each i we set $x_i = f_i$ and we set $y_F = x_i$ for $F = G_i$ and $y_{F'} = 0$ for every other $F' \in \mathcal{F}_i$. In other words we are using only one flow for each commodity i. The only non-trivial fact to check is that this solution is feasible. For this we need to verify that $\sum_i y_F \cdot F_e \leq c_e$ but this easily follows from our definition of $F = G_i$ and Constraint (4) in Figure 1. Since $x_i = \sum_{F' \in \mathcal{F}_i} y_{F'} = f_i$ for all i, we see that the two solutions have the same value.

Second, consider a feasible solution y to the packing formulation in Figure 2(b). Let x_i be the fraction of d_i routed for commodity i and for each flow $F \in \mathcal{F}_i$, y_F is the fraction routed on F with $\sum_{F \in \mathcal{F}_i} y_F = x_i$. We construct a feasible solution to the compact LP as follows. For each commodity i we set $f_i = x_i$. For each $e \in E$ and each $i \in [k]$, we set $f_{i,e} = \frac{1}{d_i} \sum_{F \in \mathcal{F}_i} F_e \cdot y_F$. Note that $f_{i,e}$ is simply scaling by d_i the total flow on e from all $F \in \mathcal{F}_i$. Since each $F \in \mathcal{F}_i$ is a flow of d_i units from s_i to t_i and $\sum_{F \in \mathcal{F}_i} y_F = x_i$ we see that $f_{i,e}$, $e \in E$, corresponds to sending a total fraction x_i of d_i units of flow from s_i to t_i. We focus on Constraints (4) and (5) in Figure 1. We observe that $\sum_i d_i f_{i,e} = \sum_i d_i \frac{1}{d_i} \sum_{F \in \mathcal{F}_i} F_e \cdot y_F = \sum_i \sum_{F \in \mathcal{F}_i} y_F \cdot F_e$ and the last term is at most c_e from the feasiblity of given solution for the packing formulation. This proves that Constraint (4) in Figure 1 is satisfied for the solution we constructed. We observe that for each $F \in \mathcal{F}_i$ and each $e \in E$ we have $F_e \leq c_e$ since F is a feasible flow in the capacities. Thus $F_e/d_i \leq c_e/d_i$ and since $y_F \geq 0$ for each $F \in \mathcal{F}_i$, we have $\sum_{F \in \mathcal{F}_i} y_F \cdot F_e/d_i \leq c_e/d_i \sum_{F \in \mathcal{F}_i} y_F$ which implies that $f_{i,e} d_i \leq f_i c_e$. Thus the solution also satisfies (5) in Figure 1. This finishes the proof of the equivalence.

Hence, the proofs and results in Section III that lead to Theorem 3.5 and Corollary 5.1 also apply to a randomized rounding approach based on the compact formulation, as we explain in Section IV.

III. Solving the Packing LP Relaxation

In this section, we show how to round a (fractional) solution to the primal packing ANF LP relaxation to get our $(1 - \epsilon, \min\{k, \log n / \log\log n\})$-approximation guarantees, for any $1/m \leq \epsilon < 1$. Given the equivalence of the two ANF LP formulations, and the fact that the set of feasible solutions of the compact LP can be viewed as a subset of the feasible solutions of the packing LP, all the approximation guarantees in this section also hold if we were to round an optimal solution of the LP relaxation of the compact formulation, as we will do in Section IV and in our implementations.

We first show that there exists a polynomial-time algorithm for directly solving the packing LP, in spite of its exponential size. This will be useful since it will allow us to use a multiplicative weight update (MWU) approach for efficiently approximating the packing LP (Section V), which will in turn allow us to solve the interesting extensions of the ANF problem we present in Section VI in practice.

A. Solving the Packing LP in Polynomial Time

It is not at first obvious that the LP relaxation of the ANF MIP can be solved in polynomial time. There are two ways to see why this is indeed possible. One would be to simply solve the *equivalent compact (polynomial-size) edge-flow formulation*; another would be to show via the Ellipsoid method that the dual has an *efficient separation oracle*. In this section we focus on the latter, which gives us a more flexible formulation that can also handle the extensions in Section VI.

$$\begin{aligned} \min \sum_{e \in E} c_e \ell_e + \sum_{i=1}^{k} z_i & \\ z_i + \sum_{e \in E} F_e \ell_e \geq w_i & \qquad 1 \leq i \leq k, F \in \mathcal{F}_i \\ \ell_e \geq 0 & \qquad e \in E \\ z_i \geq 0 & \qquad 1 \leq i \leq k \end{aligned}$$

Fig. 3. Dual of the LP relaxation for ANF.

In Figure 3, we present the dual LP to the formulation in Figure 2(b). Note that the LP relaxation in Figure 2(b), has already been simplified, for convenience, by projecting out the variables x_i. There are two types of variables in the dual: First, for each of the capacity constraints, we associate a dual variable ℓ_e and for each constraint limiting the total flow to 1 we associate a dual variable z_i. (Recall that the value F_e is a constant and not a variable.)

The following lemma shows that one can use a polynomial-time separation oracle for solving the dual LP.

Lemma 3.1: There is a polynomial-time separation oracle for the dual LP.

Proof: The dual LP is easily seen to reduce to s-t minimum-cost flow. Given non-negative values for the variables $\ell_e, e \in E$ and $z_i, 1 \leq i \leq k$ we compute the minimum-cost flow for each pair (s_i, t_i) of d_i units with edge costs given by $\ell_e, e \in E$. Let this cost be q_i. The values are feasible for the dual if and only if $z_i + q_i \geq w_i$ for $1 \leq i \leq k$. If there is an i for which $z_i + q_i < w_i$ the corresponding minimum cost flow F for pair i defines the violated constraint. The minimum-cost flow problem is solvable in polynomial time and hence there is a polynomial-time separation oracle for the dual LP. □

Hence, standard techniques allow one to solve the primal LP from an optimum solution to the dual LP in polynomial time. However, since it would be impractical to solve the dual LP using the Ellipsoid method, we present and implement an efficient MWU algorithm for solving the ANF packing LP.

B. Rounding the Packing LP Relaxation

In this section, we show how to round a (fractional) solution of the primal packing ANF LP relaxation to get our $(1 - \epsilon, \min\{k, \log n / \log\log n\})$-approximation, for any $1/m \leq \epsilon < 1$. We will need the following standard Chernoff bound [28]:

Theorem 3.2: Let $X_1, \ldots, X_n$ be n independent random variables (not necessarily distributed identically), with each variable X_i taking a value of 0 or v_i for some value $0 < v_i \leq 1$. Let $X = \sum_{i=1}^{n} X_i$ be their sum. Then the following hold:

- *For $\mu \geq E[X]$ and $\delta > 0$, $\Pr[X \geq (1+\delta)\mu] < \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^\mu$.*
- *For $0 \leq \mu \leq E[X]$ and $\delta \in (0,1)$, $\Pr[X \leq (1-\delta)\mu] < e^{-\delta^2 \mu/2}$.*

Randomly rounding a feasible solution to the LP relaxation is straightforward, and is very similar to the standard rounding via the path formulation for the Maximum Edge Disjoint Problem (MEDP) pioneered in the work of Raghavan and Thompson [31]. We consider the support of the solution to the LP relaxation: For each i, the LP relaxation identifies some h_i flows $F_{i_1}, F_{i_2}, \ldots, F_{i_{h_i}} \in \mathcal{F}_i$ along with non-negative values $y_{F_{i_1}}, \ldots, y_{F_{i_{h_i}}}$ such that their sum is at most 1. The

randomized algorithm simply picks for each i independently, at most one of the flows in its support where the probability of picking F_{i_j} is exactly $y_{F_{i_j}}$. Note that the probability that one chooses to route pair i is exactly $\sum_{F \in \mathcal{F}_i} y_F \leq 1$.

We will analyze the algorithm with respect to the weight of the LP solution $\sum_{i=1}^{k} w_i \sum_{j=1}^{h_j} y_{F_{i_j}}$. We refer to this quantity as W_{LP}. We refer to the value of an optimum solution to the LP as OPT_{LP} and the value of an optimum integer solution as OPT_{IP}. We observe that $\text{OPT}_{\text{LP}} \geq \text{OPT}_{\text{IP}}$ and $\text{OPT}_{\text{LP}} \geq W_{\text{LP}}$. Note that when solving the formulation in Figure 2(b) or the compact LP relaxation of the formulation in Figure 1, the LP solution obtained will be optimal and hence $W_{\text{LP}} = \text{OPT}_{\text{LP}}$; however, the solution obtained via the multiplicative-weight update algorithm of Section V may only approximate OPT_{LP} and hence one could indeed have $\text{OPT}_{\text{LP}} > W_{\text{LP}}$. We will also assume that $\text{OPT}_{\text{LP}} \geq w_{\max}$, since we can discard from consideration any commodity i that cannot be routed alone in the network, as it will never be part of a feasible solution of the MIP formulation, and hence $w_{\max} \leq \text{OPT}_{\text{IP}} \leq \text{OPT}_{\text{LP}}$.

The following two lemmas bound the throughput approximation and the edge capacity violation ratio of the algorithm:

Lemma 3.3: Let Z be the (random) weight of the pairs chosen to be routed by the algorithm. Then $E[Z] = W_{LP}$ and $\Pr[Z < (1-\delta)W_{LP}] < e^{-\frac{\delta^2}{2}\frac{W_{LP}}{w_{\max}}}$. In particular, $\Pr[Z < (1-\delta)W_{LP}] < e^{-\delta^2/2}$.

Proof: Let Y_i be the indicator for pair i being chosen to be routed. We have $Z = \sum_{i=1}^{k} w_i Y_i$. The rounding algorithm implies that $\Pr[Y_i = 1] = \sum_{F \in \mathcal{F}_i} y_F$. Hence, by linearity of expectation, $E[Z] = \sum_i w_i E[Y_i] = \sum_i w_i \sum_{F \in \mathcal{F}_i} y_F = W_{\text{LP}}$. Let $Z_i = \frac{w_i}{w_{\max}} Y_i$ and let $Z' = \sum_i Z_i$. Note that $Z_i \leq 1$ and $\sum_i Z_i = \frac{1}{w_{\max}} Z$. Since Z' is a sum of independent random variables, each of which is in $[0,1]$, we can apply the lower-tail Chernoff bound for Z', and obtain a lower-tail bound for Z:

$$\begin{aligned}\Pr[Z < (1-\delta)W_{\text{LP}}] &= \Pr[Z' < (1-\delta)W_{\text{LP}}/w_{\max}] \\ &= \Pr[Z' < (1-\delta)E[Z']] \leq e^{-\frac{\delta^2}{2}(W_{\text{LP}}/w_{\max})}\end{aligned}$$

□

Lemma 3.4: For $m \geq 9$ and $b > 1$ the probability that the total flow on an edge e is more than $(3b \ln m / \ln \ln m)c_e$ is at most $e^{-1.5\ b \ln m - \frac{3b \ln b \ln m}{\ln \ln m} - 1}$. Via the union bound, the probability that the total flow on any *edge e is more than $(3b \ln m / \ln \ln m)c_e$ is at most $e^{-(1.5\ b-1)\ln m - \frac{3b \ln b \ln m}{\ln \ln m} - 1}$.*

Proof: Let X_e be the random variable indicating the total flow on edge e. Let $X_{e,i}$ be the flow on e from the flow chosen for pair i. We have $X_e = \sum_{i=1}^{k} X_{e,i}$ and moreover the variables $X_{e,i}$, $1 \leq i \leq k$, are independent by the algorithm. Note that $0 \leq X_{e,i} \leq c_e$ since each flow in $\mathcal{F}_i$ is valid by definition. Further

$$E[X_e] = \sum_i E[X_{e,i}] = \sum_{i=1}^{k} \sum_{F \in \mathcal{F}_i} F_e \cdot y_F \leq c_e.$$

We now apply the Chernoff bound to see that $\Pr[X_e > (3b \ln m / \ln \ln m)c_e] \leq \frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}$ where $(1+\delta) = 3b \ln m / \ln \ln m$; we note that the standard bound has all variables bounded in $[0,1]$ while all our variables are in $[0, c_e]$ but we can simply scale all variables by c_e. We have $\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}} = e^{\delta - (1+\delta)\ln(1+\delta)}$. We consider the expression $\delta - (1+\delta)\ln(1+\delta)$, where $(1+\delta) = 3b \ln m / \ln \ln m$:

$$\begin{aligned}&\delta - (1+\delta)\ln(1+\delta) = \\ &= (1+\delta)(1 - \ln(1+\delta)) - 1 \\ &= (3b \ln m / \ln \ln m)(1 - \ln(3b \ln m / \ln \ln m))) - 1 \\ &= (3b \ln m / \ln \ln m)(1 - \ln 3 - \ln b - \ln \ln m + \ln \ln \ln m) - 1 \\ &\leq (3b \ln m / \ln \ln m)(-\ln b - \frac{1}{2} \ln \ln m) - 1 \\ &\leq -1.5\ b \ln m - 3b \ln b \ln m / \ln \ln m - 1.\end{aligned}$$

In the above, we used the fact that $\ln \ln m - \ln \ln \ln m \geq \frac{1}{2} \ln \ln m$ for $m \geq 9$.

The second part follows easily via the union bound over all the m edges. □

We can now put together the preceding lemmas to derive our bicriteria approximation. We will henceforth assume that $m \geq 9$. Let S be the random set of pairs routed by the algorithm. Let $\mathcal{E}_1$ be the event that $w(S) < (1-\delta)W_{\text{LP}}$. From Lemma 3.3, $\Pr[\mathcal{E}_1] \leq e^{-\delta^2/2}$. Let $\mathcal{E}_2$ be the event that there is some edge e such that the flow on e is more than $(3b \ln m / \ln \ln m)c_e$. From Lemma 3.4, $\Pr[\mathcal{E}_2] \leq e^{-1.5\ b \ln m - 3b \ln b \ln m / \ln \ln m - 1}$. For $b = 1$ and $m \geq 9$ we see that $\Pr[\mathcal{E}_2] \leq e^{-4.29} < 0.0138$. Choosing $\delta = 1/2$, $\Pr[\mathcal{E}_1] \leq 0.8825$. Thus $\Pr[\mathcal{E}_1 \text{ or } \mathcal{E}_2] \leq 0.9$. This implies that with probability at least 0.1, the set S of routed pairs satisfies the property that $w(S) \geq 0.5 W_{\text{LP}}$ *and* the congestion of every edge is at most $3 \ln m / \ln \ln m$. In other words, if $W_{\text{LP}} = \text{OPT}_{\text{LP}} \leq \text{OPT}_{\text{IP}}$, we obtain a $(1/2, 3 \ln m / \ln \ln m)$-bicriteria approximation with probability at least 0.1. One can boost the success probability by repetition. If the rounding is repeated $10c \ln m$ times, then with probability at least $1 - (1-0.1)^{10c \ln m} \geq 1 - m^{-c}$ (in other words with high probability), one of the rounded solutions is a $(1/2, 3 \ln m / \ln \ln m)$-bicriteria approximation

We now refine the preceding argument to show that the quality of the rounded solution can get arbitrarily close to W_{LP} but with lower probability, and examine the trade-off required in the congestion and number of repetitions required. Suppose we want $w(S) \geq (1-\epsilon)W_{\text{LP}}$ for some small $0 < \epsilon < 1/2$. Let $\mathcal{E}_1$ be the event that this does not happen. From Lemma 3.3, we have that $\Pr[\mathcal{E}_1] \leq e^{-\epsilon^2/2}$. Let $\mathcal{E}_2$ be the event that some edge congestion exceeds $3b \ln m / \ln \ln m$. Lemma 3.4 allows us to upper bound this probability. Suppose we choose b such that $\Pr[\mathcal{E}_2] \leq \epsilon^2/6$. Then

$$\begin{aligned}&\Pr[\bar{\mathcal{E}}_1 \cap \bar{\mathcal{E}}_2] = 1 - \Pr[\mathcal{E}_1 \cup \mathcal{E}_2] \geq -\Pr[\mathcal{E}1] - \Pr[\mathcal{E}_2]) \\ &\geq 1 - e^{-\epsilon^2/2} - \epsilon^2/6 \geq \epsilon^2/6.\end{aligned}$$

This would yield a $(1-\epsilon, 3b \ln m / \ln \ln m)$ bicriteria approximation with probability at least $\epsilon^2/6$ and one can boost this via repeating $O(\frac{1}{\epsilon^2} \ln m)$ times to get the approximation with high probability. Thus it remains to estimate b such that $\Pr[\mathcal{E}_2] \leq \epsilon^2/6$. From Lemma 3.4, it suffices to choose b such that $(1.5b - 1)\ln m + 3b \ln b \ln m / \ln m \ln m + 1 \geq \ln(6/\epsilon^2)$. In particular, it suffices to have $b \geq c\frac{\ln(1/\epsilon)}{\ln m}$ for some fixed constant c. Thus for all $\epsilon \geq 1/m$, a fixed constant b (e.g., $b = 1.85$) suffices! Note however that the number of repetitions grows as $\Omega(1/\epsilon^2)$ to guarantee a good solution with high probability. This completes the proof of our main theorem:

Theorem 3.5: For $m \geq 9$ and any $1/m \leq \epsilon < 1$, there is a polynomial-time randomized algorithm that yields a $(1-\epsilon, O(\ln m / \ln \ln m + \ln(1/\epsilon)/\ln m))$-approximation with high probability. Moreover, by setting $\epsilon = 1/m$, we guaran-

Algorithm 1 Randomized Rounding Algorithm

Input: Directed graph $G(V, E)$ with edge capacities $c_e > 0, \forall e \in E$; set of k pairs of commodities (s_i, t_i), each with demand $d_i \geq 0$ and weight $w_i \geq 0$; $\epsilon \in (0, 1]$.
Output: The final values of f_i and $f_{i,e}$ and $\sum w_i f_i$.

1: Let $\tilde{f}_i, \tilde{f}_{i,e}, \forall i \in [k], \forall e \in E$, be a feasible solution to compact LP.
2: For each $i \in [k]$, independently, set $f_i = 1$ with probability $\tilde{f}_i$, otherwise set $f_i = 0$.
3: Rescale the fractional flow $\tilde{f}_{i,e}$ from the LP solution on edge e for commodity i by $\frac{1}{\tilde{f}_i}$: I.e., $f_{i,e} = \frac{\tilde{f}_{i,e}}{\tilde{f}_i} \cdot f_i$ and the flow for commodity i on e is given by $f_{i,e} d_i$.
4: If $\sum_i w_i f_i \geq (1-\epsilon) \sum w_i \tilde{f}_i$ and $\sum_i f_{i,e} d_i \leq (3b \ln m / \ln \ln m) c_e$ for all $e \in E$, return the corresponding flow assignments given by f_i and $f_{i,e}, \forall i \in [k]$ and $e \in E$. Otherwise, repeat steps 2 and 3, $O((\ln m)/\epsilon^2)$ times.

tee a $(1 - 1/m, O(\ln m / \ln \ln m))$*-approximation with high probability.*

Noting that it is trivial to get a $(1, k)$-approximation by simply routing all the commodities at full demand, we get the following corollary, stating our full approximation guarantees:

Corollary 5.1: For $m \geq 9$ *and any* $1/m \leq \epsilon < 1$*, there is a polynomial-time randomized algorithm that yields a* $(1-\epsilon, \min\{k, O(\ln m / \ln \ln m)\})$*-approximation with high probability.*

For completeness, we describe a different rounding scheme in [8] using an *alteration approach*, that may also be of interest in certain settings and gives a better tradeoff in terms of repetitions.

IV. Randomized Rounding of the Compact LP

In this section, we describe the randomized rounding algorithm that we will use in our simulations. Algorithm 1 performs randomized rounding on the k total flow variables of the compact LP and can be viewed as a simpler and more efficient case of the randomized rounding algorithm outlined in Section III-B. Another advantage of Algorithm 1 is that it leads to a surprisingly simple derandomized algorithm.

A. Randomized Rounding Algorithm

We use randomized rounding to round the total fraction $\tilde{f}_i$ of d_i that the compact LP routes for commodity i to $f_i = 1$, with probability $\tilde{f}_i$, and to 0 otherwise. If we set f_i to 1, then in order to satisfy flow conservation constraints (i.e., Constraint (3) of Figure 1), we need to re-scale all the $\tilde{f}_{i,e}$ values by $1/\tilde{f}_i$, obtaining the flows $f_{i,e}$ (if $f_i = 0$ then $f_{i,e} = 0$, for all $e \in E$). We repeat Steps 2-3 of Algorithm 1 $\Theta((\ln m)/\epsilon^2)$ times or until we obtain the desired $((1-\epsilon), 3b \ln m / \ln \ln m)$-approximation bounds, amplifying the probability of getting a desired outcome.

Given the equivalence that we showed between the packing and the compact LP, which implied among other things that the two LPs have optimal solutions of the same value and that Algorithm 1 corresponds to the packing randomized rounding approach described in Section II-B when restricted to the subset of solutions to the compact LP, we get the following corollary to Theorem 3.5:

Corollary II.6.2: Algorithm 1, when run on an optimal solution to the compact LP, achieves a $((1-\epsilon), \min\{k, 3b \ln m / \ln \ln m\})$-approximation *for the ANF problem on* arbitrary directed networks *with high probability, for a suitable constant* $b > 1/m$*, e.g.* $b = 1.85$*, and any* $1/m \leq \epsilon < 1$.

B. An Efficient Derandomized Algorithm

In this section, we show how to derandomize Algorithm 1. Our derandomized algorithm is particularly attractive for its simplicity and efficiency in practice (see Section VII), unlike most other derandomized algorithms in the literature, whose implementations are cumbersome and ineffective. Our deterministic algorithm leverages the method of pessimistic estimators first introduced by Raghavan [30] to efficiently compute conditional expectations, which will guide the construction of the (α, β)-approximate solution.

We first introduce the following notation. Let $z_i = 0$ if Algorithm 1 has not selected commodity i to be routed, and let $z_i = 1$ if i was admitted. Now, let $\text{fail}(z_1, \ldots, z_k) \rightarrow \{0, 1\}$ denote the failure function of not constructing an (α, β)-approximate solution, i.e., $\text{fail}(z_1, \ldots, z_k) = 1$ if and only if the constructed solution either does not achieve an α-fraction of the LP's (weighted) throughput or the capacity of some edge is exceeded by a factor larger than β. We use Z_i to denote the $\{0, 1\}$-indicator random variable for whether commodity i is routed in one execution of Steps 3-4 of Algorithm 1, i.e., $\Pr[Z_i = 1] = \tilde{f}_i$ and $\Pr[Z_i = 0] = 1 - \tilde{f}_i$. We have shown in Section III-B that $E[\text{fail}(Z_1, \ldots, Z_k)] < 1$ holds (cf. Theorem 3.5), implying the existence of an (α, β)-approximate solution. Given the above definitions, we employ the following notation to denote the conditional expectation of a function $f : \{0, 1\}^k \rightarrow \{0, 1\}$:

$$E[f(z_1, \ldots, z_i, Z_{i+1}, \ldots, Z_k)] = \Pr[f(Z_1, \ldots, Z_k) = 1 \mid Z_1 = z_1, \ldots, Z_i = z_i].$$

As computing $E[\text{fail}(z_1, \ldots, z_i, Z_{i+1}, \ldots, Z_k)]$ is generally computationally prohibitive, we will now derive a pessimistic estimator $\text{est} : \{0, 1\}^k \rightarrow \mathbb{R}_{\geq 0}$, such that the following holds for all $i \in [k]$ and all $(z_1, \ldots, z_i) \in \{0, 1\}^i$:

Upper Bound $E[\text{fail}(z_1, \ldots, z_i, Z_{i+1}, \ldots, Z_k)] \leq E[\text{est}(z_1, \ldots, z_i, Z_{i+1}, \ldots, Z_k)]$

Efficiency $E[\text{est}(z_1, \ldots, z_i, Z_{i+1}, \ldots, Z_k)]$ can be computed efficiently.

Furthermore, the estimator's value must initially be strictly less than 1 for the derandomization:

Base Case $E[\text{est}(Z_1, \ldots, Z_k)] < 1$ holds initially.

In the following, we discuss how such a pessimistic estimator is used to derandomize the decisions made by Algorithm 1 before introducing the actual estimator $\text{est}^{\alpha}_{\beta}$ in Lemma 4.1. Algorithm 2 first computes an LP solution just as Algorithm 1, but then uses the pessimistic estimator to guide its decision towards deterministically constructing an approximate solution. Specifically, each commodity is either routed or rejected such that the conditional expectation $E[\text{est}^{\alpha}_{\beta}(z_1, \ldots, z_i, Z_i, \ldots, Z_n)]$ is minimized. Given that initially $E[\text{est}^{\alpha}_{\beta}(Z_1, \ldots, Z_k)] < 1$, this procedure terminates with a solution $(z_1, \ldots, z_k)$ such that the failure function $\text{fail}(z_1, \ldots, z_k)$ is strictly upper bounded by 1. Specifically, $1 > E[\text{est}^{\alpha}_{\beta}(Z_1, \ldots, Z_k)] \geq E[\text{est}^{\alpha}_{\beta}(z_1, Z_2, \ldots, Z_k)] \geq \ldots \geq E[\text{est}^{\alpha}_{\beta}(z_1, \ldots, z_k)]$ is guaranteed and therefore, for the binary function fail, $\text{fail}(z_1, \ldots, z_k) = 0$ must hold. Furthermore, the algorithm is efficient (i.e., runs in polynomial time) as long as $\text{est}^{\alpha}_{\beta}$ can be evaluated in polynomial time.

Algorithm 2 Deterministic Approximation for ANF

Input: Directed graph $G(V,E)$ with $c_e > 0, \forall e \in E$ and $m \geq 9$; (s_i,t_i), $d_i, w_i \geq 0, \forall i \in [k]$; $\alpha = 1 - 1/m, \beta = 3b\ln m/\ln\ln m$, $b = 1.85$; estimator $\text{est}^{\alpha}_{\beta} : \{0,1\}^k \to \mathbb{R}_{\geq 0}$.
Output: The final values of f_i and $f_{i,e}$ and $\sum w_i f_i$

Compute optimal solution $\vec{\tilde{f}}$ to compact edge-flow LP.
Let $Z_i \in \{0,1\}$ be s.t. $\Pr[Z_i = 1] = \tilde{f}_i$ and $\Pr[Z_i = 0] = 1 - \tilde{f}_i$, for $i \in [k]$.
Compute fail_est $\leftarrow E[\text{est}^{\alpha}_{\beta}(Z_1, \ldots, Z_{i-1}, Z_i, \ldots, Z_n)]$.
for each $i \in [k]$ **do**
  if $E[\text{est}^{\alpha}_{\beta}(z_1, \ldots, z_{i-1}, 0, Z_{i+1}, \ldots, Z_n)] <$ fail_est **then**
    Set $z_i \leftarrow 0$.
  else
    Set $z_i \leftarrow 1$.
    Update fail_est $\leftarrow E[\text{est}^{\alpha}_{\beta}(z_1, \ldots, z_i, Z_{i+1}, \ldots, Z_n)]$.
return solution given by $\vec{z}$: if $z_i = 1$, then $f_i = 1$ and $f_{i,e} = \tilde{f}_{i,e}/\tilde{f}_i$, else $f_i = f_{i,e} = 0$, $\forall i \in [k]$.

Lemma 4.1 introduces the specific pessimistic estimator $\text{est}^{\alpha}_{\beta}$ for which the above three correctness criteria (upper bound, efficiency, base case) are proven. We assume $\alpha = 1 - 1/m$ and $\beta = 3b\ln m/\ln\ln m$, for $m \geq 9$ and $b = 1.85$.

Lemma 4.1 (Pessimistic Estimator): The function est^{α}_{β} is a pessimistic estimator for the ANF:

$$est^{\alpha}_{\beta}(Z_1,\ldots,Z_k) = est_{\alpha}(Z_1,\ldots,Z_k) + \sum_{(u,v)\in E} est^{(u,v)}_{\beta}(Z_1,\ldots,Z_k),$$

where $est_{\alpha}(Z_1,\ldots,Z_k) = e^{-\theta_{\alpha}(1-\delta_{\alpha})\tilde{\mu}} \prod_{l\in[k]} E[e^{\theta_{\alpha} Z_i \frac{w_i}{w_{\max}}}]$,

with $\delta_{\alpha} = \dfrac{1}{m}$, $\tilde{\mu} = \dfrac{w_{LP}}{w_{\max}}$, *and* $\theta_{\alpha} = \ln(1-\delta_{\alpha})$;

and $est^{(u,v)}_{\beta}(Z_1,\ldots,Z_k)$

$= e^{-\theta_{\beta}(1+\delta_{\beta})\hat{\mu}} \prod_{l\in[k]} E[e^{\theta_{\beta} Z_i \frac{(f_{i,(u,v)}/\tilde{f}_i)}{c_{(u,v)}}}]$,

with δ_{β}

$= \dfrac{3b\ln m}{\ln\ln m - 1}$, $b = 1.85$, $\hat{\mu} = 1$, *and* $\theta_{\beta} = \ln(1+\delta_{\beta})$.

For the proof of Lemma 4.1, we restate Theorem 3.2 as Theorem 4.2, since we will need the intermediate inequalities (a) and (c):

Theorem 4.2 ([28]): Let X be the sum of k random variables $X_1,\ldots,X_k$ with $X_l \in [0,1]$ for $l \in [k]$, and let $\mu_l = E[X_l]$. If $\hat{\mu}_l \geq \mu_l$, for each $l \in [k]$, the following holds for any $\delta > 0$ with $\theta = \ln(\delta+1)$ and $\hat{\mu} = \sum_{l\in[k]} \hat{\mu}_l$:

$$Pr[X \geq (1+\delta)\cdot\hat{\mu}] \overset{(a)}{\leq} e^{-\theta\cdot(1+\delta)\cdot\hat{\mu}} \cdot \prod_{l\in[k]} E[e^{\theta\cdot X_l}] \overset{(b)}{\leq} \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\hat{\mu}}$$

Moreover, if $\tilde{\mu}_l \leq \mu_l$, for each $l \in [k]$, the following holds for any $\delta \in (0,1)$ with $\theta = \ln(1-\delta)$ and $\hat{\mu} = \sum_{l\in[k]} \hat{\mu}_l$:

$$Pr[X \leq (1-\delta)\cdot\tilde{\mu}] \overset{(c)}{\leq} e^{-\theta\cdot(1-\delta)\cdot\tilde{\mu}} \cdot \prod_{l\in[k]} E[e^{\theta\cdot X_l}] \overset{(d)}{\leq} e^{-\delta^2\cdot\tilde{\mu}/2}$$

Proof: [Proof of Lemma 4.1] The following three properties are to be shown: (i) upper bound, (ii) efficiency, and (iii) base case. We first discuss properties (i) and (iii).

Corollary II.6.2 showed that the probability of obtaining an (α,β)-approximate solution via randomized rounding is at least $1/(6\cdot m^2)$. To obtain this result, a union bound argument was employed, which used probabilistic bounds on not achieving at least an α fraction of the optimal throughput or on exceeding the capacity of some edge by a factor of β.

For the throughput, we apply the first part of Theorem 4.2, while for each edge's capacity violation, we apply the second part of Theorem 4.2, deriving the following pessimistic estimators: Estimator est_{α} is obtained from the application of the bounds in Theorem 4.2 within the proof of Lemma 3.3. Specifically, the application of the Chernoff bound in Lemma 3.3 yields the following — restated over the variables Z_i — with the parameters δ_{α}, θ_{α} and $\tilde{\mu}$ as specified above. The middle expression below directly yields the throughput pessimistic estimator.

$$\Pr[\sum_{l\in[k]} w_l\cdot Z_l < \alpha\cdot w_{LP}] \leq e^{-\theta_{\alpha}\cdot(1-\delta)\cdot\tilde{\mu}} \cdot \prod_{i\in[k]} E[e^{\theta_{\alpha}\cdot Z_i\cdot w_i/w_{\max}}] \leq e^{-1/(2\cdot m^2)}$$

The estimator $\text{est}^{(u,v)}_{\beta}$ is analogously obtained by applying the bounds in Theorem 4.2 within Lemma 3.4 for each edge $(u,v) \in E$. Specifically, for a single edge (u,v), the following is obtained, where again, the middle expression is used to obtain the pessimistic estimator $\text{est}^{(u,v)}_{\beta}$ for $(u,v) \in E$.

$$\Pr[\sum_{i\in[k]} f_{i,(u,v)} > \beta\cdot c_{(u,v)}] \leq e^{-\theta_{\beta}\cdot(1+\delta_{\beta})\cdot\hat{\mu}} \cdot \prod_{i} E[e^{\theta_{\beta}\cdot Z_i\cdot f_{i,(u,v)}/c_{(u,v)}}] \leq \frac{1}{6\cdot m^2}$$

Revisiting the union bound argument, we obtain that $\text{est}^{\alpha}_{\beta}$ indeed yields an upper bound on the failure probability to construct an (α,β)-approximate solution, and that initially $E[\text{est}^{\alpha}_{\beta}(Z_1,\ldots,Z_k)] \leq 1 - 1/(6\cdot m^2) < 1$ holds for $m \geq 9$. This shows that properties (i) and (iii) are satisfied.

Considering the efficiency property, we note the following. Both est_{α} and $\text{est}^{(u,v)}_{\beta}$ consist of products of k factors, where the expectations for different commodities can be computed independently. Due to the binary nature of the variables Z_i, these expectations can be computed in constant time. □

Given the above outlined derandomization process and the correctness of the pessimistic estimator in Lemma 4.1, the following main theorem of this section is obtained.

Theorem 4.3: Using est^{α}_{β} as a pessimistic estimator, Algorithm 2 is a deterministic (α,β)*-approximation algorithm for the ANF problem with $\alpha = 1 - 1/m$ and $\beta = 3b\ln m/\ln\ln m$, with $b = 1.85$, for $m \geq 9$.*

V. MWU Algorithm

While the compact edge-flow LP relaxation can always be solved in polynomial time, one may run into space issues when attempting to solve it in practice: The disadvantage of using a standard LP solver to solve the compact edge-flow LP relaxation is that the number of variables is km which is quadratic in the input size, and the number of constraints is m. Standard LP solvers often require space proportional to km^2 which can be prohibitive even for moderate instances (since it is almost cubic in the input size). One advantage of the packing LP formulation over the compact formulation is that one can use well-known multiplicative weight update (MWU)-based Lagrangean relaxation approaches to obtain a $(1-\gamma)$-approximation, for any $0 < \gamma < 1$. Although the

$$(a) \quad \max \sum_{i=1}^{k} w_i \sum_{F \in \mathcal{F}_i} y_F$$

$$\sum_{i=1}^{k} \sum_{F \in \mathcal{F}_i} F_e \cdot y_F \leq c_e \qquad e \in E$$

$$y_F \geq 0 \qquad F \in \mathcal{F}_i, 1 \leq i \leq k.$$

$$(b) \quad \min \sum_{e \in E} c_e \ell_e$$

$$\sum_{e \in E} F_e \ell_e \geq w_i \qquad 1 \leq i \leq k, F \in \mathcal{F}_i$$

$$\ell_e \geq 0 \qquad e \in E$$

Fig. 4. (a) LP Relaxation with no constraint on total amount routed per commodity; (b) its dual.

convergence time can be slow depending on the accuracy required, the space requirement for the MWU approach is $O(k + m)$, which is linear in the input size. In addition, there are several optimization heuristics based on the MWU algorithm that can result in very efficient implementations in practice. Since the MWU framework is standard, we only describe and explain the algorithm here and state the known guarantees on the number of iterations and time complexity, referring the reader to standard treatments in the literature [5] for a formal analysis on the correctness guarantees.

In our implementations, we will also run Algorithm 1 starting from the fractional solution output obtained from the MWU algorithm, which only guarantees a $(1 - \gamma)$ approximation on the throughput of the LP relaxation for $\gamma \in (0, 1)$. In that case, we let f_i in Algorithm 1 be the total fraction of commodity i routed by the MWU algorithm and $\tilde{f}_{i,e}$ be the fraction of the flow for commodity i going through edge e—namely, $\tilde{f}_i = \bar{f}_{i,(s'_i,s_i)}$ and $\tilde{f}_{i,e} = \bar{f}_{i,e}$, for all e, where s'_i and the variables $\bar{f}_{i,e}$'s are as defined in Algorithm 3. Hence the throughput approximation guarantee for Algorithm 1 when rounding the MWU solution will be $(1 - \epsilon)(1 - \gamma)$.

A. Algorithm Description

MWU-based algorithms are iterative and provide a way to obtain arbitrarily good relative approximation algorithms for a large class of linear programs such as packing, covering and mixed packing and covering LPs. In particular we can apply it to the packing LP in Fig 2(b). The LP has two types of packing constraints, one involving the capacities, and the other involving the total amount of flow routed for each commodity. It is useful to simplify the LP further in order to apply a clean packing framework and so we alter the given graph $G = (V, E)$ as follows. For each given demand pair (s_i, t_i) we add a dummy source s'_i and connect it to s_i with an edge (s'_i, s_i) of capacity equal to d_i. We replace the pair (s_i, t_i) with the pair (s'_i, t_i), which ensures that the total amount of flow for the pair is at most d_i, and further allows us to eliminate the first set of constraints in Figure 2(b). In the modified instance, we hence only have edge capacity constraints and the problem becomes a pure maximum throughput problem that does not limit the total flow for each commodity. The dual LP also simplifies in a corresponding fashion. These are shown in Figure 4.

The MWU Algorithm 3 solves the primal LP in Fig 4 in an iterative fashion as follows. It takes as input an error parameter $\gamma \in (0, 1)$ and its goal is to output a feasible solution of value at least $(1-\gamma)$ times the optimum LP solution value. Note that the primal LP has an exponential number of variables but only m non-trivial constraints corresponding to the edges, so it maintains only an implicit representation of the primal variables. The MWU algorithm can be viewed as a primal-dual algorithm as well and as such it maintains "weights" (hence the name multiplicative weights update) for each edge e, which together correspond to the dual variables ℓ_e. To avoid confusion with the weights of commodities we use the term lengths. The algorithm maintains lengths $\ell_e, e \in E$ which are initialized to 1. The algorithm roughly maintains the invariant that ℓ_e is exponential in the current total flow $\bar{F}_e$ on edge e; more formally, for a parameter $\eta = \ln m/\gamma$ the algorithm maintains the invariant that $\ell_e \simeq \exp(\eta \bar{F}_e/c_e)$ where $\bar{F}_e$ is the total flow on e. In each iteration the goal is to find a good commodity/pair to route. To this end, the algorithm computes for each commodity (s_i, t_i) a minimum-cost s_i-t_i flow of d_i units where the cost on e is equal to ℓ_e. Let this cost be $\rho(i)$. It then chooses the commodity i^* that has the smallest $\rho(i)/w_i$ ratio among all pairs, as the currently best commodity to route. The algorithm then routes a small amount for i^* along the minimum cost flow computed in that iteration. This corresponds to the step size δ which is chosen to be sufficiently small (but not too small) to ensure the correctness of the algorithm. After routing the flow for i^*, the lengths on the edges are updated to reflect the increase in flow on the edges. The algorithm proceeds in this fashion for several iterations until termination. One can terminate using several different criteria while ensuring correctness. Here we stop the algorithm if we route a commodity with the given step size and it violates some edge capacity.

Algorithm 3 MWU for Multi-Commodity ANF Problem

Input: Directed graph $G(V, E)$, $c : E \to \mathbb{R}^+$, a set S of k commodity pairs (s_i, t_i), each with demand d_i and $\gamma \in \mathbb{R}^+$.
Output: Total flow $\bar{F}_e$, for each edge e, and flows $\bar{f}_{i,e}$ on each edge e for each commodity i

Change G by adding dummy terminal s'_i and edge (s'_i, s_i) with capacity d_i. This ensures that we don't route more than d_i units for pair i. We will assume this has been done and simply use (s_i, t_i) instead of (s'_i, t_i).
Define a length/cost function $\ell : E \to \mathbb{R}^+$, initialize $\ell_e \leftarrow 1, \forall e \in E$
Define a function $\bar{F} : E \to \mathbb{R}_{\geq 0}$, initialize $\bar{F}_e \leftarrow 0, \forall e \in E$
Let $\bar{f}_{i,e} \leftarrow 0$ be the fractional flow assignment for commodity i on edge e, for all $i \in S, e \in E$
Define $\eta \leftarrow \frac{\ln |E|}{\gamma}$
repeat
  for each commodity $i \in S$ **do**
    Compute min-cost flow of d_i units from s_i to t_i with capacities c_e and cost given by ℓ (if no feasible flow then pair i can be dropped). Let this flow be defined by $G_{i,e}, e \in E$ and let cost of this flow be $\rho(i) = \sum_e \ell_e G_{i,e}$
  Set $i^* \leftarrow \text{argmin}_{i \in S} \frac{\rho(i)}{w_i}$
  Compute $\delta \leftarrow \min_e \frac{\gamma}{\eta} \cdot \frac{G_{i^*,e}}{c_e}$
  for each e **do**
    if $\bar{F}_e + \delta G_{i^*,e} > c_e$ **then**
      Output $\bar{F}$ and $\bar{f}_{i,e}, \forall e \in E, \forall i \in S$ and halt
  Update $\bar{F}_e \leftarrow \bar{F}_e + \delta G_{i^*,e}, \forall e \in E$
  Update $\bar{f}_{i^*,e} \leftarrow \bar{f}_{i^*,e} + \delta G_{i^*,e}/d_{i^*}, \forall e \in E$
  Update $\ell_e \leftarrow \exp(\eta \bar{F}_e/c_e), \forall e \in E$
until termination

B. Analysis of Iterations, Run-Time and Space

The algorithm's running time is dependent on the time to compute a minimum-cost flow and on the total number of iterations. It is known that the MWU algorithm,

as suggested above, terminates in $O(m \log m/\gamma^2)$ iterations. Each iteration requires computing k minimum-cost flows. Many algorithms are known for minimum-cost flow ranging from strongly polynomial-time algorithms to polynomial-time scaling algorithms, as well as practically fast algorithms based on network-simplex. Instead of listing these, we can upper bound the run-time by $O(\text{MCF}(n,m)km \log m/\gamma^2)$ where $\text{MCF}(n,m)$ is the min-cost flow algorithm's running time on a graph with n nodes and m edges. In terms of space, we observe that the algorithm only maintains, for each edge, the total flow and length on the edge, and the total flow routed for each commodity: This is $O(k+m)$. The algorithm also needs space to compute a minimum-cost flow and that depends on the algorithm used for it. Most algorithms for minimum-cost flow use space near-linear in the input graph.

The algorithm as described above is a plain "vanilla" implementation of the general MWU algorithm. As such the running time is rather high and computing k minimum-cost flows in each iteration is expensive. Several optimizations can be done from both a theoretical and a practical point of view. We do not discuss these issues in detail since this is not the main focus of this paper. We develop a simple heuristic – the *permutation routing* heuristic – based on these ideas that has also theoretical justification, and that will be discussed and used for the simulations in Section VII.

VI. Potential Problem Extensions

The packing formulation for the ANF was introduced in Section II-B together with a simple randomized rounding algorithm. Besides the practical tractability established in Section V, the proposed packing framework for the ANF has further advantages. Specifically, it can be easily adapted to cater for problem extensions such as when flows are restricted to k-splittable flows, must obey fault-tolerance criteria, or are restricted to shortest paths.

In the following we describe some of these extensions and how the packing formulation may be adapted together with the separation procedure. Notably, some problem extensions allow for compact LP formulations, however, casting the problems in terms of the packing formulation is generally less complex and therefore helps in establishing whether a problem extension can be efficiently approximated in the first place.

Henceforth, our goal is to solve the maximum throughput problem in the all-or-nothing model while restricting the nature of flows that are allowed for each commodity. The ANF allows flow for each commodity to be split in arbitrary ways while unsplittable flow requires all the flow for a commodity to use a single path. However, there are several intermediate settings of interest, and other constraints, that occur in practice. Recall that in setting up the formulation in Figure 2, $\mathcal{F}_i$ for each pair i is the set of valid s_i-t_i flows in G. This is a large implicit set, and the way we solve the LP relaxation is via the separation oracle. The separation oracle corresponds to finding a minimum-cost flow from $\mathcal{F}_i$ given some edge lengths/costs. The MWU algorithm can be viewed as an efficient, albeit approximate, way to solve the large implicit LP relaxation via the separation oracle. Moreover, once the LP is solved, the randomized rounding step picks one of the flows per commodity. This flexibility allows us to solve the LP and round even when $\mathcal{F}_i$ is restricted in some fashion. We outline a few extensions that can be addressed via this framework.

Integer flows: Recall that in ANF we allow splittable flows. However in some settings it is useful to have flow for each commodity on each edge to be integer valued; here we assume that d_i is an integer for each i. In order to handle this we can set $\mathcal{F}_i$ to be the set of all integer s_i-t_i flows. Now the min-cost flow routine needs to find an integer flow between s_i and t_i of d_i units. This is easy to ensure since there always exists an integer valued min-cost flow as long as the demands and the edge capacities are all integer valued. We reduce each c_e to $\lfloor c_e \rfloor$ without loss of generality.

Splitting into a small number of paths: In some applications it is important that the flow for each pair is not split by too much. How do we quantify this? One way is to consider h-splittable flows where h is an integer parameter. This means that flow for each pair can be decomposed into at most h paths. When $h = 1$ we obtain unsplittable flow and if we set $h = |E|$ we obtain ANF. We can handle the special case where each of the h flow paths has to be used to send the same amount of flow which is d_i/h. For this purpose we define $\mathcal{F}_i$ to be the set of all such flows. To compute a min-cost flow in $\mathcal{F}_i$ we simply need to find a min-cost flow of h units from s_i to t_i in the graph with capacities adjusted as follows: for each edge e with capacity c_e we change it to $\lfloor h \cdot c_e/d_i \rfloor$. Baeier et al. [6] considered this maximum throughput problem, however, they only considered uni-criteria approximation algorithms and provided a reduction to the unsplittable case; the approximation ratios that one can obtain without violating capacities are very poor while our focus here is on bicriteria approximation that achieve close to optimum throughput.

Fault-tolerance and routing along disjoint paths: In some settings the flow for a pair (s_i, t_i) needs to be fault-tolerant to edge and/or node failures. There are several ways this is handled in the networking literature. One common approach is to send the flow for each commodity along h disjoint paths, each carrying d_i units. This can be handled by an approach very similar to the preceding paragraph where we compute min-cost flow on h disjoint paths; note that in the preceding paragraph the h paths could share edges. Another approach to fault-tolerance is to use what are called h-route flows [1], [25]. One can find a min-cost h-route flow in polynomial time [1]. Hence, one can also use the framework to maximize throughput while each routed pair uses an h-route flow.

Using few edges or short paths: We now consider the setting when the flow for a commodity is required to use a small number of edges or the flow has to be routed along paths with small number of hops. These constraints not only arise in practice but also help improve the theoretical bounds on congestion. One can show that if each flow uses only d edges then the bicriteria approximation can be improved; the congestion required for a constant factor approximation becomes $O(\log d/\log\log d)$ rather than $O(\log m/\log\log m)$; for single paths the analysis can be seen from [7] and we can generalize it to our setting. Suppose we wish to route flow for each commodity whose support consists only of some given number h of edges. As above we need to solve for min-cost s_i-t_i flow that satisfies this extra constraint. However this additional constraint is no longer so easy to solve and in some cases can be NP-Hard. However, if one allows for a constant factor relaxation for the number of edges h, and an additional constant factor in the edge congestion, one can address this more complex constraint by using linear programming based ideas (see [12] for an example).

VII. Simulation Results

In this section we study the performance of our approximation algorithms for the ANF problem on real-world networks.

Our proof of concept computational evaluation is meant to provide general guidelines about the relative efficacy of the algorithms in terms of the *achieved throughput approximation factor* α and the *edge capacity violation ratio* β. The achieved throughput approximation ratio is taken as the solution obtained by the run divided by the optimal LP solution (which is a lower bound on the exact approximation ratio based on the optimal MIP solution rather than its LP relaxation). Notably, due to the bi-criteria nature of our approximations with solutions being allowed to exceed edge capacities (by at most a factor of β), solutions may yield empirical throughput approximation factors of $\alpha > 1$.

Beyond analyzing the performance of our randomized rounding and derandomized algorithms, we also investigate the impact of varying the methodology by which the LP is solved. Specifically, we study the performance of solving the compact LP formulation directly; of solving the multiplicative weight update algorithm (MWU); and of solving the MWU-based Permutation Routing (PR) heuristic described below. While the run-time of our prototypical MWU implementation generally exceeds the run-time of solving the compact LP formulation using a commercial solver, our MWU implementation serves as a proof of concept of its practical applicability and will also enable the extensions outlined in Section VI, which depend on the packing formulation. In addition, we remark that MWU may be useful for larger networks in practice (such as for the random network instance that we address in Section VII-C), as it does not suffer from the same space complexity limitations as solving the compact LP via standard LP solvers.

Note that the simulation results for the prior state-of-the-art algorithm for constant-throughput approximations for the ANF problem [26]—originally designed to handle uniform demands, edge capacities and weights—have been reproduced in this paper when running the randomized rounding algorithm with the compact edge-flow LP, since this algorithm is in essence the same as the algorithm in [26], now adapted to handle non-uniform demands, edge capacities and weights (in addition to some fine tuning optimizations). Our theoretical approximation results in this paper actually also validate the simulation results in [26], since the simulations in [26] already suggested that the edge capacity violations incurred by randomized rounding based on the compact edge-flow LP were logarithmic (and not polynomial as the theoretical guarantees of [26] suggested).

A. Permutation Routing Heuristic

Without proper optimization, the run-time of the MWU algorithm can be slow due to the computation of k minimum-cost flows as a separate procedure in each iteration. As a practical solution, we introduce a heuristic based on Algorithm 3 that provides a significant reduction in computational cost, while still yielding solutions comparable to those by MWU in practice. We refer to this as the *Permutation Routing (PR)* algorithm. In the following we outline how this new algorithm differs from the original MWU algorithm and we refer the reader to Algorithm 4 for the complete pseudocode description.

Our algorithm is motivated by theoretical algorithms for maximum throughput packing problems in the *online* arrival model and the *random arrival order* models. It is known that for packing problems, in the random arrival model, one can obtain arbitrarily good performance compared to the offline optimal solution if the resource requirements of the arriving items (these correspond to flows in our setting) are sufficiently small when compared to the capacities [3], [22], [24]. The analytical ideas are related to online learning and MWU.

Algorithm 4 Permutation Routing Algorithm

Input: $\gamma \in \mathbb{R}^+$, Directed Graph $G(V, E)$, $c : E \to \mathbb{R}^+$, a set S of k pairs of commodities (s_i, t_i) each with demand d_i, weight w_i, an estimate Est of the optimal fractional ANF solution for (G, S)

Output: Total flow $\bar{F}_e$, $\forall e \in E$, and fractional flow assignment $\bar{f}_{i,e}$, $\forall i \in S, e \in E$

Change G by adding dummy terminal s_i' and edge (s_i', s_i) with capacity d_i. This ensures that we don't route more than d_i units for pair i. We will assume this has been done and simply use (s_i, t_i) instead of (s_i', t_i).
Initialize an empty flow $\bar{F}_e \leftarrow 0, \forall e \in E$
Set $\eta \leftarrow \frac{\ln |E|}{\gamma}$
Set $r \leftarrow \frac{\ln |E|}{\gamma^2}$
Let $\bar{f}_{i,e} \leftarrow 0$ be the fractional flow assignment for commodity i on edges e for all $i \in S, e \in E$
Define edge costs $\ell_e = 1, \forall e \in E$
Make r copies of the k commodities of S and let A be a list of these rk commodity pairs.
Let $B = (x_1, y_1), \ldots, (x_{rk}, y_{rk})$ be a random permutation of A.
for $j = 1$ to rk **do**
    Let $(s_i, t_i), i \in S$ be the commodity associated with pair (x_j, y_j).
    Compute min-cost flow of d_i units from s_i to t_i with edge costs defined by ℓ and obtain flow assignment F' and solution cost $\rho = \sum_{e \in E} \ell_e F_e'$.
    Compute $\tau = \sum_{e \in E} \ell_e c_e$.
    if $\frac{w_i}{\rho} \geq \frac{Est}{\tau}$ and $\bar{F}_e + \frac{F_e'}{r} \leq c_e, \forall e \in E$ **then**
        Update $\bar{F}_e \leftarrow \bar{F}_e + \frac{F_e'}{r}, \forall e \in E$
        Update $\bar{f}_{i,e} \leftarrow \bar{f}_{i,e} + \frac{1}{r}, \forall e \in E$
        Update $\ell_e \leftarrow \exp\left(\frac{\eta \cdot \bar{F}_e}{c_e}\right), \forall e \in E$
Output $\bar{F}_e$ and $\bar{f}_{i,e}$, $\forall e \in E, \forall i \in S$

We develop our heuristic as follows: Recall that we are seeking a fractional solution. We take each commodity pair i with demand d_i and split it into r "copies," each with a demand of d_i/r. Here r is a sufficiently large parameter to ensure the property that d_i/r is "small" compared to the capacities. From the MWU analysis, and also the analysis in random arrival order models, one sees that $r = \Omega(\ln m/\gamma^2)$ suffices. Given the k original commodity pairs, we create $k \cdot r$ total pairs from the copies. We now randomly permute these pairs and consider them one-by-one. When considering a pair, the algorithm evaluates the "goodness" of the pair in a fashion very similar to that of the MWU algorithm. It maintains a length for each edge that is exponential in its current loads, and computes a minimum cost flow for the current pair (note that the pair's demand is only a $1/r$ fraction of its original demand); it accepts this pair if the cost of the flow is favorable compared to an estimate of the optimum solution. If it accepts the pair, it routes its entire demand (which is the $1/r$'th fraction of the original demand). Otherwise this pair is rejected and never considered again. Thus the total number of minimum cost flow computations is $O(k \cdot r)$ when compared to $O(k \cdot m \cdot \log m/\gamma^2)$ in the MWU algorithm. As mentioned above, a worst-case theoretical analysis requires $r = \Omega(\log m/\gamma^2)$ to guarantee a $(1-\gamma)$-approximation, however, in practice a smaller value of

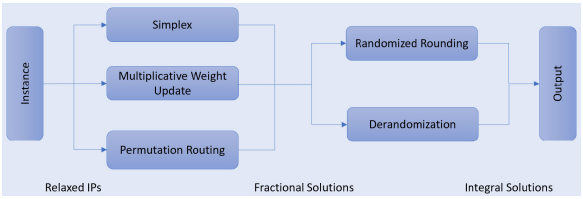


Fig. 5. Flowchart showing our experimental methodology. A directed path from instance to output defines an *experiment*.

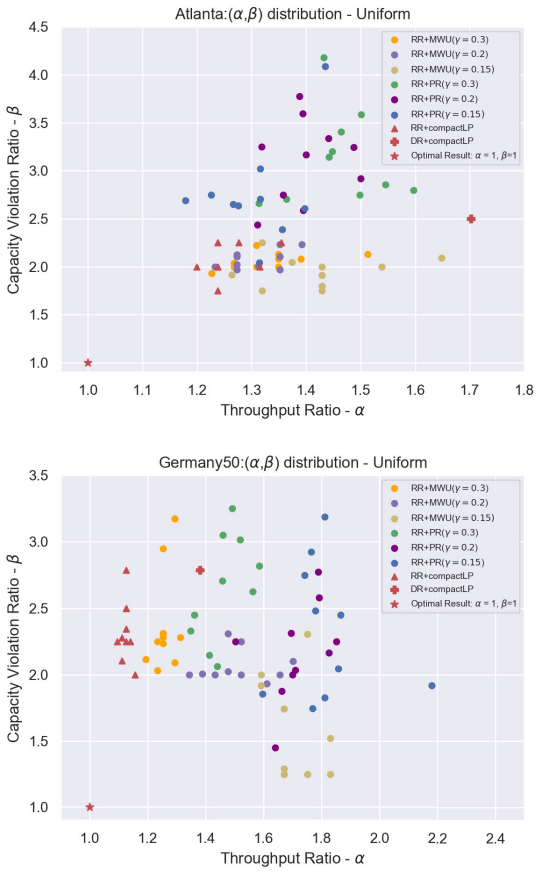


Fig. 6. Experimental results on the Atlanta and Germany50 networks with uniform edge capacities, weights and demands.

r can be chosen. Note that an original pair (s_i, t_i) with demand d_i is routed to a fraction r_i/r where r_i is the number of copies of i that are admitted by the random permutation algorithm. The algorithm requires an estimate of the optimum solution which can be obtained via binary search or other methods.

B. Methodology

We now describe the problem instances and the implementations of our approximation algorithms. Our code is publicly available at [9].

1) Problem Instances: Following [26], we study real-world networks together with corresponding real-world source-sink pairs obtained from the survivable network design library (SNDlib) [29]. Our choice of networks from the SNDlib is given in Table I, covering several general scenarios, e.g. a small network with large number of commodities, or a dense network with low number of commodities. We randomly perturb the uniform weights, demands and edge capacities of the chosen networks in SNDlib to test our algorithms' ability to accommodate variable weights and demands on networks with varying edge capacities. Namely, we independently chose

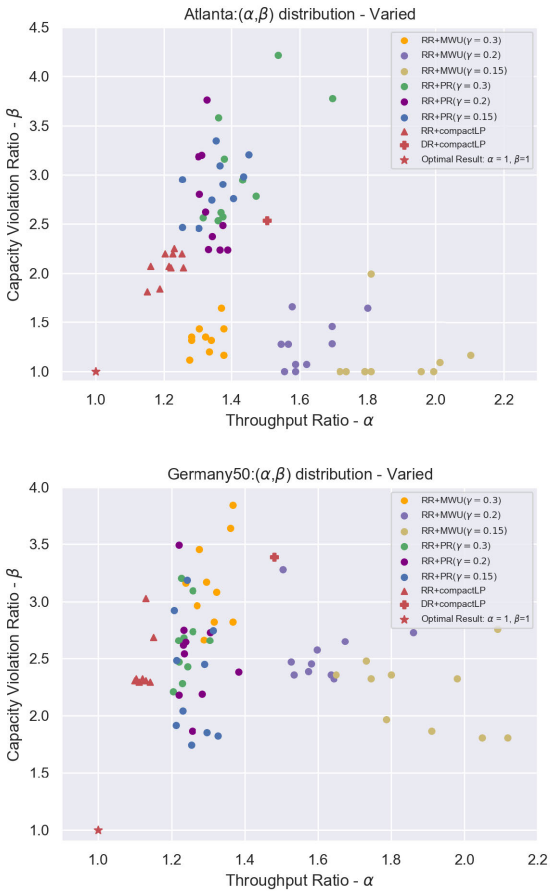


Fig. 7. Experimental results on the Atlanta and Germany50 networks with varied edge capacities, weights and demands.

TABLE I

LIST OF STUDIED INSTANCES FROM SNDLIB [29] WHERE n IS THE NUMBER OF NODES, m IS THE NUMBER OF EDGES, k IS THE NUMBER OF COMMODITIES IN THE ORIGINAL NETWORK AND k' IS THE NUMBER OF ROUTABLE COMMODITIES IN THE VARIED NETWORK

Network	n	m	k	k'	General Description
Atlanta	15	44	210	193	Small network, high commodity count
Germany50	50	176	662	627	Sparse network, high commodity count
Di-yuan	11	84	22	22	Dense network, low commodity count
Dfn-gwin	11	94	110	107	Dense network, high commodity count

uniformly random edge capacities from 20 to 60, commodity demands from 25 to 75, and commodity weights from 1 to 10 (the benchmark SNDlib data has all edge capacities at 40, demands at 50, and weights at 1). After these changes, we find that only a fraction of the given commodities can be satisfied if routed alone in the network: We discard any commodity that cannot be routed on its own and let k' denote the number of remaining commodities.

2) Algorithms: We have implemented both the randomized and derandomized rounding algorithms detailed in Sections IV-A and IV-B on an Apple M1 processor with 16GB RAM. We solve the compact formulation via CPLEX V22.1.1 and approximately solve the packing LP via the MWU algorithm or the faster permutation routing heuristic. We choose $\epsilon = \frac{1}{9}$ and $b = 1.85$ in Algorithms 1 and 2, implying a target throughput approximation factor of $\alpha \geq 1 - \epsilon = \frac{8}{9}$ and target edge capacity violation ratio of $\beta \leq 3b \ln m / \ln \ln m = 5.55 \ln m / \ln \ln m$, where m is the number of network edges, for the algorithms. More specifically, for

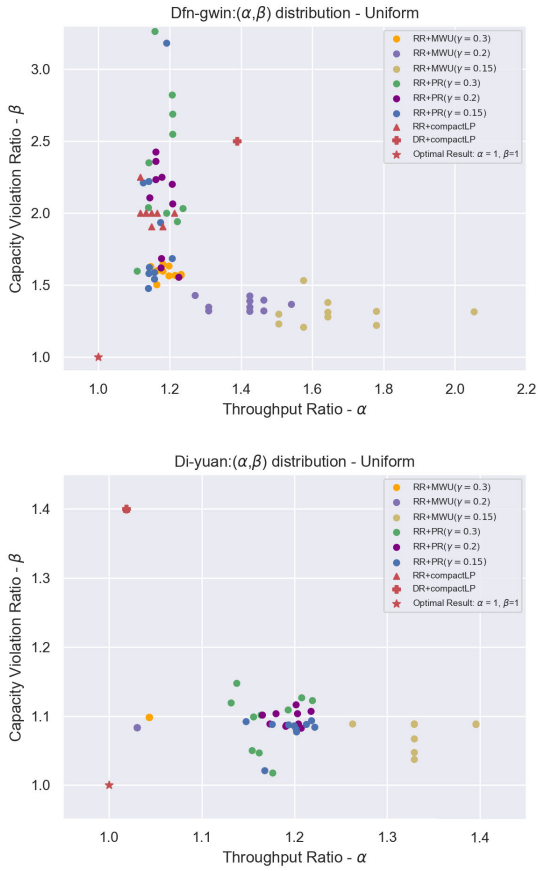


Fig. 8. Experimental results on the Dfn-gwin and Di-yuan networks with uniform edge capacities, weights and demands.

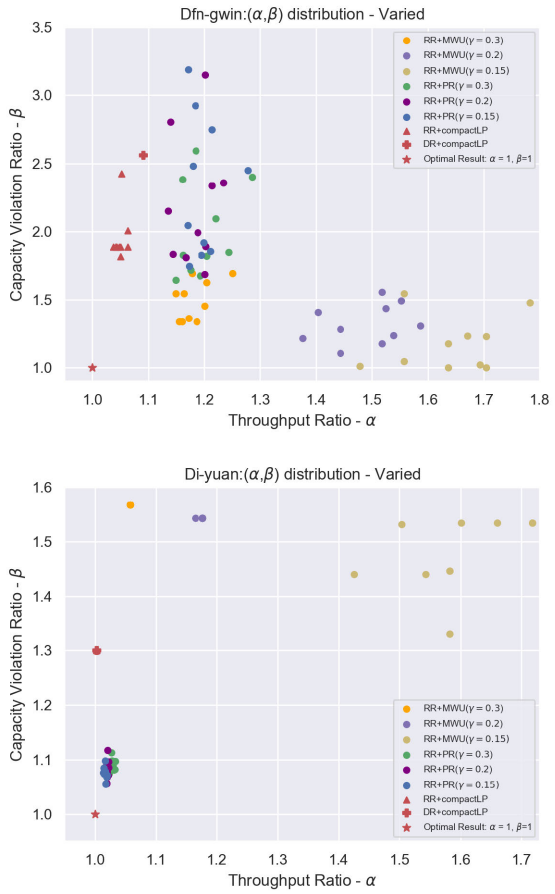


Fig. 9. Experimental results on the Dfn-gwin and Di-yuan networks with varied edge capacities, weights and demands.

the Atlanta and Germany50 networks, we target edge capacity violations $\beta \leq 15.78$ and 17.47, respectively.

We define an *experiment* as the execution of a higher level algorithm (either randomized or derandomized rounding) in concert with an LP-solving subroutine (CPLEX for compact LP or our MWU and PR implementations) on a particular network. We summarize our methodology through the flowchart in Figure 5. For an experiment that includes randomized rounding, we execute this algorithm 10 times to obtain a total of 10 different samples per experiment. For each of these 10 executions, 100 rounds of rounding are recorded and of these rounded solutions, we report on the solution of highest throughput whose capacity violations lie below our theoretical bounds. We consider three different γ values, namely 0.15, 0.2, and 0.3, to study performance vs. run-time trade-offs of the MWU algorithm and the PR heuristic. Due to the slow convergence of MWU, we introduce speed-up mechanisms where (i) during any iteration, if the post-update smallest mincost flow solution is not at least 50 percent larger than the pre-update smallest min-cost flow solution, then we do not recompute this in the subsequent iteration, and (ii) the maximum number of iterations is capped at 10,000.

C. Experimental Results

In this section, we expand upon our computational results. We first focus our attention on the performance of two sparse networks, Atlanta and Germany50, and then move to two smaller but comparatively denser networks, Di-yuan and Dfn-gwin. We report results in terms of the achieved throughput factor α, edge capacity violation factor β and wall-clock run-times.

Our experiments are summarized visually in two kinds of plots. The qualitative plots in Figures 6–9 show the empirical throughput and edge capacity violation ratios obtained by executions of the various algorithms for each network separately. Note that we report on 10 data points when applying randomized rounding in contrast to the single data point for the derandomized algorithm using the compact LP solution. In the figures, we use RR to denote randomized rounding and DR to denote the derandomized algorithm, followed by the specific LP used (compact LP, MWU or PR) and choice of γ for MWU and PR. For reference, we include a red star data point indicating the optimal LP solution, which corresponds to $\alpha = \beta = 1$. In Figures 10–13, we show how the run-times compare among different algorithms.

Figures 6 summarizes the results on networks Atlanta and Germany50 under the default uniform weights, demands and edge capacities given by [29]. This figure also replicates experiments from [26], though here we additionally test Algorithms 3 and 4 in conjunction with Algorithm 1. We also test the modified form of the original networks—where we randomly perturb the weights, demands and edge capacities as described in Section VII-B—in Figure 7. Figures 8 and 9 present experimental results on the additional networks from [29], namely Dfn-gwin and Di-yuan, respectively for uniform and varied weights, demands and edge capacities. Note that in Figures 8–9, the values for the compact LP–based randomized rounding and the derandomized solution for the Di-yuan network have very close values of α and β and hence overlap in the plots.

The compact LP solution combined with both the randomized and derandomized algorithms produces in general

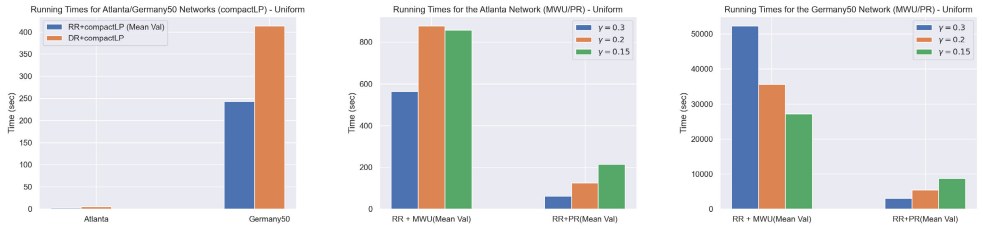


Fig. 10. Run-times for uniform Germany50 and Atlanta networks.

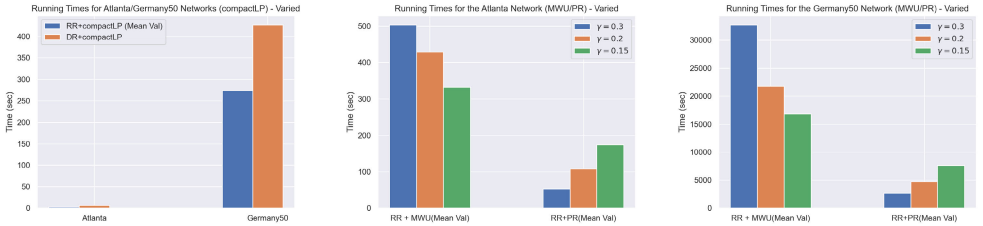


Fig. 11. Run-times for varied Germany50 and Atlanta networks.

$\beta \leq 2.5$, which is much smaller than our established theoretical bounds. We see noticeably larger values of α and marginally larger values of β with the derandomized algorithm (at the expense of a higher run-time). With respect to MWU-based randomized rounding solutions, the values of α and β obtained are concentrated around their means. For the PR subroutines, we observe more variance over the parameter space, and we typically see much higher capacity violations without a significant gain in throughput. The value of PR over MWU shows up in the run-time plots, but PR still loses to the compact LP. We expand on our run-time discussions next. As for the impact of γ on the quality of the solutions obtained using MWU and PR, the general trend is that the lower the γ, the lower the edge capacity violation β with an increase in α.

Figures 10–13 show that the compact LP-based randomized rounding run-times are at least two orders of magnitude faster than the implementations using MWU and at least one order of magnitude faster than those using PR in general, with the exception of Dfn-gwin with varied network parameters. The varied Dfn-gwin run-times for RR+PR (for all γ's) are in fact better than the run-times for solving the compact LP with CPLEX and then applying randomized rounding. The randomized rounding algorithm outperforms our derandomized algorithm based on the compact LP solution (see the leftmost plots in Figures 10-13). We believe this to be in part due to our naive implementation of the pessimistic estimators, which does not cache intermediate results.

Regarding the performance of the permutation routing algorithm, we see a run-time decrease by a factor of at least one half compared to the "vanilla" MWU algorithm, while slightly compromising on the generally less favorable higher capacity violations, for any network and any γ. For the MWU algorithm, we expect in general to see the run-time increase as γ decreases, however, due to our speed-up mechanisms, the opposite may be true. This is attributed to the fact that with a smaller γ, the increase in flow is likewise smaller, and thus the updates to edge costs are smaller, implying that the threshold for skipping min-cost flow calculations is met more often. Thus, the run-time is reduced for smaller values of γ, though at the expense of throughput approximation quality.

We further report on our implementations of the derandomized algorithm in conjunction with MWU and PR. In general, the run-times of the derandomized algorithm based on an MWU solution produced very similar graphs to what we have for the run-times of the randomized rounding approach based on MWU in Figures 10-13. With respect to the approximation bounds on throughput and edge capacity violation, the derandomized approach based on MWU or PR produced a much wider set of values, with, for example, $\alpha = 34.2$ and $\beta = 17.5$ for PR, and $\alpha = 10.7$ and $\beta = 5.3$ for MWU, both for $\gamma = 0.15$ on the Germany network with uniform demands. In general, the derandomized algorithm with MWU or PR produced significantly higher values of α and β for the Atlanta, Germany and Dfn-gwin networks (both for the uniform and varied capacities and demands cases) when compared to the respective runs with the randomized rounding approach, while the MWU and PR derandomized results for the Di-yuan network were more comparable to the respective randomized rounding results. See [8] for additional plots, with the simulation results for all instances of the derandomized algorithm in conjunction with MWU and PR.

To validate the need for approximation algorithms for the ANF problem in practice, we tried to directly solve the compact MIP formulation (Figure 1(a)) for the different uniform network instances we considered. While the smaller networks such as Atlanta and Di-yuan finished within a minute, the Germany50 network failed to terminate after 24 hours. For comparison, we were able to solve the compact LP relaxation for the Germany50 network in less than 240 seconds. For completeness, the MIP solutions for both Atlanta and Di-yuan routed 21 commodities, while the compact LP solution had an optimal value of 26.84 and 21.6, respectively.

Concluding, we see our results as a first step towards efficiently approximating the ANF and its potential extensions. While randomized rounding wins in terms of run-time, the deterministic rounding generally achieves slightly higher throughputs at the expense of higher edge capacity violations. Furthermore, while solving the compact LP is shown to be much quicker in practice, the proposed MWU algorithm will render tackling the problem extensions of Section VI tractable and our proposed permutation routing heuristic can in practice substantially reduce run-times. With respect to space, it is least efficient to solve the compact LP directly. In fact, it may be impossible to do so if the network instance is very large. The MWU and Permutation Routing algorithms rely on repeated and discarded computations of single commodity min-cost

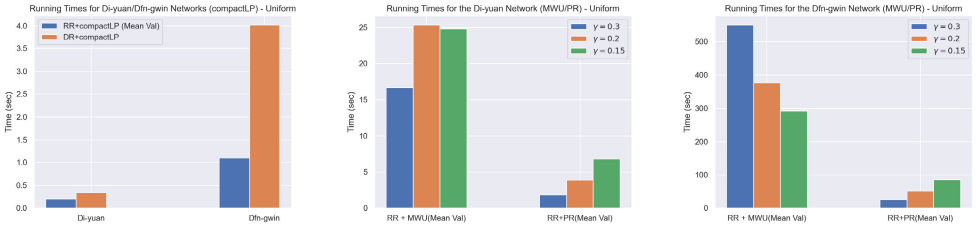


Fig. 12. Run-times for uniform Di-yuan and Dfn-gwin networks.

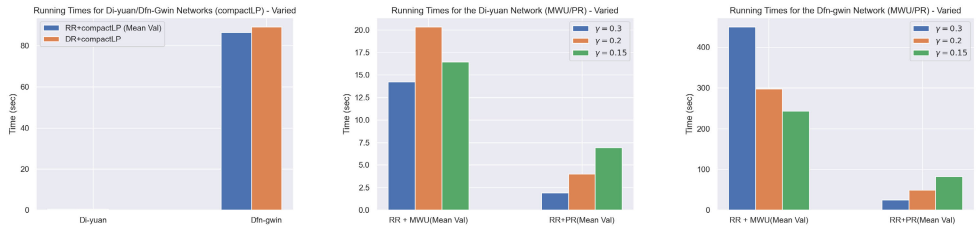


Fig. 13. Run-times for varied Di-yuan and Dfn-gwin networks.

flow, whose corresponding LPs exhibit much fewer constraints than ANF for the same network instances. To illustrate this, we generated a random network with $n = 449$, $m = 15856$, and $k = 48$: While solving the compact LP with CPLEX crashed due to memory errors for this instance, we ran MWU for up to 1500 iterations and did not observe any memory issues.

VIII. Conclusion

We presented a novel and significantly improved bi-criteria approximation of the maximum throughput routing problem for all-or-nothing multiple commodities with arbitrary demands, which we paired with a proof of concept on efficient implementations of our algorithms in practice. We showed that our packing framework is very flexible and may hence be of interest beyond the specific model considered in this paper, e.g., in scenarios where flows should only be split into a small number of paths or use few edges. In future research, it would be interesting to develop improved rounding approaches, e.g., using resampling ideas from the Lovasz-Local-Lemma, to explore the additional applications introduced by our packing framework, as well as to study opportunities for algorithm engineering, further improving the performance of our algorithms in practice.

References

[1] C. C. Aggarwal and J. B. Orlin, "On multiroute maximum flows in networks," *Networks*, vol. 39, no. 1, pp. 43–52, Jan. 2002.

[2] O. A. Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing Amazon EC2 spot instance pricing," *ACM Trans. Econ. Comput.*, vol. 1, no. 3, pp. 1–20, Sep. 2013.

[3] S. Agrawal and N. R. Devanur, "Fast algorithms for online stochastic convex programming," in *Proc. 26th Annu. ACM-SIAM Symp. Discrete Algorithms*, Oct. 2015, pp. 1405–1424.

[4] M. Andrews, J. Chuzhoy, V. Guruswami, S. Khanna, K. Talwar, and L. Zhang, "Inapproximability of edge-disjoint paths and low congestion routing on undirected graphs," *Combinatorica*, vol. 30, no. 5, pp. 485–520, Sep. 2010.

[5] S. Arora and S. Kale, "The multiplicative weights update method: A meta-algorithm and applications," *Theory Comput.*, vol. 8, no. 1, pp. 121–164, May 2012.

[6] G. Baier, E. Köhler, and M. Skutella, "The k-splittable flow problem," *Algorithmica*, vol. 42, nos. 3–4, pp. 231–248, Jul. 2005.

[7] A. Chakrabarti, C. Chekuri, A. Gupta, and A. Kumar, "Approximation algorithms for the unsplittable flow problem," *Algorithmica*, vol. 47, no. 1, pp. 53–78, Jan. 2007.

[8] A. Chaturvedi et al., "Improved throughput for all-or-nothing multicommodity flows with arbitrary demands," 2020, *arXiv:2005.04533*.

[9] A. Chaturvedi, C. Chekuri, A. Richa, M. Rost, S. Schmid, and J. Weber, "Improved throughput for all-or-nothing multicommodity flows with arbitrary demands," Zenodo Repository, Geneva, Switzerland, Tech. Rep. zenodo.8118453, 2023, doi: 10.5281/zenodo.8118453.

[10] A. Chaturvedi, C. Chekuri, A. W. Richa, M. Rost, S. Schmid, and J. Weber, "Improved throughput for all-or-nothing multicommodity flows with arbitrary demands," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 49, no. 3, pp. 22–27, Mar. 2022.

[11] C. Chekuri and A. Ene, "The all-or-nothing flow problem in directed graphs with symmetric demand pairs," *Math. Program.*, vol. 154, nos. 1–2, pp. 249–272, Dec. 2015.

[12] C. Chekuri and M. Idleman, "Congestion minimization for multipath routing via multiroute flows," in *Proc. 1st Symp. Simplicity Algorithms (SOSA)*, 2018, pp. 3:1–3:12.

[13] C. Chekuri, S. Khanna, and F. B. Shepherd, "Multicommodity flow, well-linked terminals, and routing problems," in *Proc. 37th Annu. ACM Symp. Theory Comput.*, May 2005, pp. 183–192.

[14] C. Chekuri, S. Khanna, and F. B. Shepherd, "The all-or-nothing multicommodity flow problem," *SIAM J. Comput.*, vol. 42, no. 4, pp. 1467–1493, Jan. 2013.

[15] J. Chuzhoy, V. Guruswami, S. Khanna, and K. Talwar, "Hardness of routing with congestion in directed graphs," in *Proc. ACM Symp. Theory Comput. (STOC)*, 2007, pp. 165–178.

[16] J. Chuzhoy, D. H. K. Kim, and R. Nimavat, "Almost polynomial hardness of node-disjoint paths in grids," in *Proc. ACM Symp. Theory Comput. (STOC)*, 2018, pp. 1220–1233.

[17] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Oper. Res.*, vol. 8, no. 1, pp. 101–111, Feb. 1960.

[18] G. Desaulniers, J. Desrosiers, and M. M. Solomon, *Column Generation*, vol. 5. Cham, Switzerland: Springer, 2006.

[19] T. Erlebach and K. Jansen, "The maximum edge-disjoint paths problem in bidirected trees," *SIAM J. Discrete Math.*, vol. 14, no. 3, pp. 326–355, Jan. 2001.

[20] C. Fuerst, S. Schmid, L. Suresh, and P. Costa, "Kraken: Online and elastic resource reservations for cloud datacenters," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 422–435, Feb. 2018.

[21] N. Garg, V. V. Vazirani, and M. Yannakakis, "Primal-dual approximation algorithms for integral flow and multicut in trees," *Algorithmica*, vol. 18, no. 1, pp. 3–20, May 1997.

[22] A. Gupta and M. Molinaro, "How the experts algorithm can help solve LPs online," *Math. Oper. Res.*, vol. 41, no. 4, pp. 1404–1431, Nov. 2016.

[23] V. Guruswami, S. Khanna, R. Rajaraman, B. Shepherd, and M. Yannakakis, "Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems," *J. Comput. Syst. Sci.*, vol. 67, no. 3, pp. 473–496, Nov. 2003.

[24] T. Kesselheim, K. Radke, A. Tönnis, and B. Vöcking, "Primal beats dual on online packing LPs in the random-order model," *SIAM J. Comput.*, vol. 47, no. 5, pp. 1939–1964, Jan. 2018.

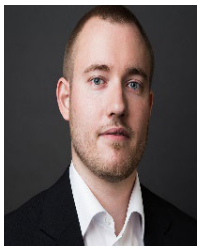
[25] W. Kishimoto, "A method for obtaining the maximum multiroute flows in a network," *Networks*, vol. 27, no. 4, pp. 279–291, Jul. 1996.
[26] M. Liu, A. W. Richa, M. Rost, and S. Schmid, "A constant approximation for maximum throughput multicommodity routing and its application to delay-tolerant network scheduling," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2019, pp. 46–54.
[27] J. C. Mogul and L. Popa, "What we talk about when we talk about cloud network performance," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 5, pp. 44–48, Sep. 2012.
[28] R. Motwani and P. Raghavan, *Randomized Algorithms*. Boca Raton, FL, USA: CRC Press, 2010.
[29] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0—Survivable network design library," *Networks*, vol. 55, no. 3, pp. 276–286, May 2010.
[30] P. Raghavan, "Probabilistic construction of deterministic algorithms: Approximating packing integer programs," *J. Comput. Syst. Sci.*, vol. 37, no. 2, pp. 130–143, Oct. 1988.
[31] P. Raghavan and C. D. Tompson, "Randomized rounding: A technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, no. 4, pp. 365–374, Dec. 1987.
[32] M. Rost, E. Döhne, and S. Schmid, "Parametrized complexity of virtual network embeddings: Dynamic & linear programming approximations," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 49, no. 1, pp. 3–10, Feb. 2019.
[33] M. Rost and S. Schmid, "Virtual network embedding approximations: Leveraging randomized rounding," in *Proc. IFIP Netw. Conf. Workshops*, May 2018, pp. 1–9.
[34] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 123–137, Sep. 2015.
[35] F. Bruce Shepherd and A. Vetta, "The inapproximability of maximum single-sink unsplittable, priority and confluent flow problems," 2015, *arXiv:1504.00627*.
[36] K. C. Webb, A. Roy, K. Yocum, and A. C. Snoeren, "Blender: Upgrading tenant-based data center networking," in *Proc. ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*, Oct. 2014, pp. 65–75.

Anya Chaturvedi was born in Kanpur, India. She received the B.Tech. degree in information technology from MNNIT Allahabad, India, in 2018, and the master's degree in computer science from Arizona State University, in 2020, where she is currently pursuing the Ph.D. degree, under the supervision of Prof. Andréa W. Richa. She worked at Intel as a Factory Automation Engineer for two years. Her research interests include design and analysis of algorithms, distributed computing, bio-inspired computing, and combinatorial optimization.

Chandra Chekuri received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology Madras in 1993 and the Ph.D. degree in computer science from Stanford University in 1998. He is currently a Professor of computer science with the University of Illinois at Urbana-Champaign (UIUC). He joined UIUC in the fall of 2006 after spending almost eight years as a Member of Technical Staff with Lucent Bell Labs. His research interests include the design and analysis of algorithms, discrete and combinatorial optimization, mathematical programming, and theoretical computer science.

Mengxue Liu received the Ph.D. degree in computer science from ASU in 2018. She has previously worked as a Research Scientist at Facebook, a Software Engineer at Snap Inc., and most recently as a Senior Software Engineer with Google.

Andréa W. Richa received the B.S. and M.S. degrees in computer science from the Federal University of Rio de Janeiro, Brazil, and the Ph.D. degree from Carnegie Mellon University. She was inducted Presidents Professor at Arizona State University (ASU) in 2022, where she is currently a Professor of computer science and engineering and an Associate Faculty with the Biodesign Institute. She joined ASU in 1998. Her research interests include distributed and network algorithms and computing in general, including bio-inspired distributed algorithms, distributed load balancing, wireless networks, and distributed hash tables. She was a recipient of the Current DoD MURI Award and the NSF CAREER Award.



Matthias Rost received the M.Sc. and Ph.D. degrees from Technische Universität Berlin, Germany, in 2014 and 2019, respectively. He is formerly a Post-Doctoral Researcher with Technische Universität Berlin. He is currently a Principal Software Engineer with Observe, Inc. His main research interests include the theoretical design of provably good algorithms for service orchestration in networks and their application in practice. He was a recipient of the KuVS Prize for his master's thesis on computing virtual aggregation and multicast trees by the German Informatics Society.

Stefan Schmid received the M.Sc. and Ph.D. degrees from ETH Zürich. He was a Postdoc Researcher at TU Munich and the University of Paderborn; a Senior Research Scientist at T-Labs, Berlin; an Associate Professor at Aalborg University, Denmark; a Full Professor at the University of Vienna, Austria; and sabbatical as a fellow at the Israel Institute for Advanced Studies (IIAS), Israel. He is currently a Professor with Technische Universität Berlin, Germany. He received the IEEE Communications Society ITC Early Career Award 2016 and the ERC Consolidator Grant 2019.

Jamison Weber is currently pursuing the Ph.D. degree in computer science with Arizona State University. His research interests include multiagent optimization, distributed computing, reinforcement learning, and randomized and approximation algorithms.