Maximum Weight Disjoint Paths in Outerplanar Graphs via Single-Tree Cut Approximators

Guyslain Naves[1], Bruce Shepherd[2], and Henry Xia[2]

[1] Aix-Marseille University, LIS, CNRS UMR 7020, guyslain.naves@univ-amu.fr
[2] University of British Columbia fbrucesh@cs.ubc.ca, h.xia@alumni.ubc.ca

Abstract. Since 1997 there has been a steady stream of advances for the maximum disjoint paths problem. Achieving tractable results has usually required focusing on relaxations such as: (i) to allow some bounded edge congestion in solutions, (ii) to only consider the unit weight (cardinality) setting, (iii) to only require fractional routability of the selected demands (the all-or-nothing flow setting). For the general form (no congestion, general weights, integral routing) of edge-disjoint paths (EDP) even the case of unit capacity trees which are stars generalizes the maximum matching problem for which Edmonds provided an exact algorithm. For general capacitated trees, Garg, Vazirani, Yannakakis showed the problem is APX-Hard and Chekuri, Mydlarz, Shepherd provided a 4-approximation. This is essentially the only setting where a constant approximation is known for the general form of EDP. We extend their result by giving a constant-factor approximation algorithm for general-form EDP in outerplanar graphs. A key component for the algorithm is to find a *single-tree* $O(1)$ cut approximator for outerplanar graphs. Previously $O(1)$ cut approximators were only known via distributions on trees and these were based implicitly on the results of Gupta, Newman, Rabinovich and Sinclair for distance tree embeddings combined with results of Anderson and Feige.

1 Introduction

The past two decades have seen numerous advances to the approximability of the maximum disjoint paths problem (EDP) since the seminal paper [17]. An instance of EDP consists of a (directed or undirected) "supply" graph $G = (V, E)$ and a collection of k *requests* (aka demands). Each request consists of a pair of nodes $s_i, t_i \in V$. These are sometimes viewed as a *demand graph* $H = (V(G), \{s_i t_i : i \in [k]\})$. A subset S of the requests is called *routable* if there exist edge-disjoint paths $\{P_i : i \in S\}$ such that P_i has endpoints s_i, t_i for each i. We may also be given a profit w_i associated with each request and the goal is to find a routable subset S which maximizes $w(S) = \sum_{i \in S} w_i$. The *cardinality version* is where we have unit weights $w_i \equiv 1$.

For directed graphs it is known [20] that there is no $O(n^{0.5-\epsilon})$ approximation, for any $\epsilon > 0$ under the assumption $P \neq NP$. Subsequently, research shifted to undirected graphs and two relaxed models. First, in the *all-or-nothing flow model* (ANF) the notion of routability is relaxed. A subset S is called routable if there is a feasible (fractional) multiflow which satisfies each request in S. In [6] a polylogarithmic approximation is given for ANF. Second, in the *congestion* model [24] one is allowed to increase the capacity of each edge in G by some constant factor. Two streams of results ensued. For general graphs, a polylogarithmic approximation is ultimately provided [10, 11, 5] with edge congestion 2. For planar graphs, a constant factor approximation is given [30, 4] with edge congestion 2. There is also an $f(g)$-factor approximation for bounded genus g graphs with congestion 3.

As far as we know, the only congestion 1 results known for either maximum ANF or EDP are as follows; all of these apply only to the cardinality version. In [23], a constant factor approximation is given for ANF in planar graphs and for treewidth k graphs there is an $f(k)$-approximation for EDP [9]. More recent results include a constant-factor approximation in the *fully planar* case where $G+H$ is planar [22, 16]. In the weighted regime, there is a factor 4 approximation for EDP in capacitated trees [8]. We remark that this problem for unit capacity "stars" already generalizes the maximum weight matching problem in general graphs. Moreover, inapproximability bounds for EDP in planar graphs are almost polynomial [12]. This lends interest to how far one can push beyond trees. Our main contribution to the theory of maximum throughput flows is the following result which is the first generalization of the (weighted) EDP result for trees [8], modulo a larger implicit constant of 224.

Theorem 1. *There is a polynomial time* 224*-approximation algorithm for the maximum weight* ANF *and* EDP *problems for capacitated outerplanar graphs.*

It is natural to try to prove this is by reducing the problem in outerplanar graphs to trees and then use [8]. A promising idea is to apply a result of [18] which gives a probabilistic embedding of an outerplanar graph G's shortest path metric into (dominating) tree metrics, which has constant distortion. This opened the possibility of finding $O(1)$-approximations for certain network

routing or design problems by using solving the problem on the much simpler tree graphs used in the embedding. The potential significance for disjoint paths is through the celebrated *Transfer Theorem* [3, 28] which established a general equivalence between low-distortion distance embeddings and low-congestion capacity embeddings. Combined, these results imply that there is a probabilistic embedding of an outerplanar graph's capacity into capacitated trees which approximates the cut capacity in the outerplanar graph up to a constant factor. One could then try to mimic the same recipe of solving for disjoint paths on the associated capacitated trees (for which an algorithm is known [6]). There is an issue however. Suppose we have a distribution on trees T_i which approximates cut capacity in expectation. We then apply the known EDP algorithm which can output a collection of request sets S_i which are routable in each T_i. While the tree embedding guarantees the convex combination of S_i's satisfies the cut condition in G, it may be that no single S_i obeys the cut condition, even approximately. This is a problem even for ANF.

We resolve the above problem by computing a **single** tree T which approximates the cuts in G – see Theorem 3. This means that any set requests which satisfies the cut condition in T, will also satisfy it in G within a constant factor. Recall that a set of requests satisfies the *cut condition* in a graph, if for each cut, the number of requests crossing the cut is at most the edge capacity of the cut. Our algorithmic proof is heavily inspired by work of Gupta [19] which gives a method for eliminating Steiner nodes in probabilistic (distance) tree embeddings for general graphs.

It turns out that having a single-tree is not enough for us and we need additional technical properties to apply the algorithm from [8]. First, our single tree T should have integer capacities and be non-expansive, i.e., $\hat{u}(\delta_T(S)) \leq u(\delta_G(S))$ (where $\hat{u}/u$ are the edge capacities in T/G and δ is used to denote the edges in the cut induced by S). To see why it is useful that T is an under-estimator of G's cut capacity, consider the classical grid example of [17]. They give an instance with a set of $\sqrt{n}$ requests which satisfy the cut condition in $2 \cdot G$, but for which one can only route a single request in the capacity of G.

If our tree is an under-estimator, then we can ultimately obtain a "large" weight subset of requests satisfying the cut condition in G itself. However, even this is not generally sufficient for (integral) routability. For a multiflow instance G/H one normally also requires that $G + H$ is Eulerian, even for easy instances such as when G is a 4-cycle. The final ingredient we use is that our single tree T is actually a **subtree** of G which allows us to invoke the following result – see Section 2.1.

Theorem 2. *Let G be an outerplanar graph with integer edge capacities $u(e)$. Let H denote a demand graph such that $G + H = (V(G), E(G) \cup E(H))$ is outerplanar. If G, H satisfies the cut condition, then H is routable in G and the routing may be computed in polynomial time.*

The key point here is that we can avoid the usual parity condition needed, such as in [25, 31, 15]. We are not presently aware of the above result's existence in the literature.

1.1 A Single-Subtree Cut Sparsifier and Related Results

Our main cut approximation theorem is the following which may be of independent interest.

Theorem 3. *For any connected outerplanar graph $G = (V, E)$ with integer edge capacities $u(e) > 0$, there is a subtree T of G with integer edge weights $\hat{u}(e) \geq 0$ such that*

$$\frac{1}{14} u(\delta_G(X)) \leq \hat{u}(\delta_T(X)) \leq u(\delta_G(X)) \text{ for each proper subset } X \subseteq V.$$

Moreover, the tree may be computer in polynomial time.

We discuss some connections of this result to prior work on sparsifiers and metric embeddings. Celebrated work of Räcke [27] shows the existence of a single capacitated tree T (not a subtree) which behaves as a flow sparsifier for a given graph G. In particular, routability of demands on T implies fractional routability in G with edge congestion $polylog(n)$; this bound was further improved to $O(\log^2 n \log\log n)$ [21]. Such single-tree results were also instrumental in an application to maximum throughput flows: a polylogarithmic approximation for the maximum all-or-nothing flow problem in general graphs [7]. Even more directly to Theorem 3 is work on cut sparsifiers; in [29] it is shown that there is a single tree (again, not subtree) which approximates cut capacity in a general graph G within a factor of $O(\log^{1.5} n \log\log n)$. As far as we know, our result is the only global-constant factor single-tree cut approximator for a family of graphs.

Räcke improved the bound for flow sparsification to an optimal congestion of $O(\log n)$ [28]. Rather than a single tree, this work requires a convex combination of (general) trees to simulate the capacity in G. His work also revealed a beautiful equivalence between the existence of good (low-congestion) distributions over trees for capacities, and the existence of good (low-distortion) distributions over trees for distances [3]. This *transfer theorem* states very roughly that for a graph G the following are equivalent for a given $\rho \geq 1$. (1) For any edge lengths $\ell(e) > 0$, there is a (distance) embedding of G into a distribution of trees which has stretch at most ρ. (2) For any edge capacities $u(e) > 0$, there is a (capacity) embedding of G into a distribution of trees which has congestion at most ρ. This work has been applied in other related contexts such as flow sparsifiers for proper subsets of terminals [14].

The transfer theorem uses a very general setting where there are a collection of valid *maps*. A map M sends an edge of G to an abstract "path" $M(e) \subseteq E(G)$. The maps may be refined for the application of interest. In the so-called *spanning tree setting*, each M is associated with a subtree T_M of G (the setting most relevant to Theorem 3). $M(e)$ is then the unique path which joins the endpoints of e in T_M. For an edge e, its *stretch* under M is $(\sum_{e' \in M(e')} \ell(e'))/\ell(e)$. In the context of distance tree embeddings this model has been studied in [2, 1, 13]. In capacity settings, the *congestion* of an edge under M is $(\sum_{e' : e \in M(e)} c(e'))/c(e)$. One can view this as simulating the capacity of G using the tree's edges with

bounded congestion. The following result shows that we cannot guarantee a single subtree with $O(1)$ congestion even for outerplanar graphs; this example was found independently by Anastasios Sidiropoulos [32].

Theorem 4. *There is an infinite family $\mathcal{O}$ of outerplanar graphs such that for every $G \in \mathcal{O}$ and every spanning tree T of G:*

$$\max_X \frac{u(\delta_G(X))}{u(\delta_T(X))} = \Omega(\log |V(G)|),$$

where the max may be taken over fundamental cuts of T.[3]

This suggests that the single-subtree result Theorem 3 is a bit lucky and critically requires the use of tree capacities different from u. Of course a single tree is sometimes unnecessarily restrictive. For instance, outerplanar graphs also have an $O(1)$-congestion embedding using a distribution of subtrees by the transfer theorem (although we are not aware of one explicitly given in the literature). This follows implicitly due to existence of an $O(1)$-stretch embedding into subtrees [18].

Finally we remark that despite the connections between distance and capacity tree embeddings, Theorem 3 stands in contrast to the situation for distance embeddings. Every embedding of the n point cycle into a (single) subtree suffers distortion $\Omega(n)$, and indeed this also holds for embedding into an arbitrary tree (where we may add extra Steiner nodes) [26].

2 Maximum Weight Disjoint Paths

In this section we prove our main result for EDP, Theorem 1. We will be using Theorem 3 which we prove in the next section.

2.1 Required Elements

We first prove the following result which establishes conditions for when the cut condition implies routability.

Theorem 2. *Let G be an outerplanar graph with integer edge capacities $u(e)$. Let H denote a demand graph such that $G + H = (V(G), E(G) \cup E(H))$ is outerplanar. If G, H satisfies the cut condition, then H is routable in G and the routing may be computed in polynomial time.*

The novelty in this statement is that we do not require the Eulerian condition on $G + H$. This condition is needed in virtually all classical results for edge-disjoint paths. In fact, even when G is a 4-cycle and H consists of a matching of size 2, the cut condition need not be sufficient to guarantee routability. The main

[3] A fundamental cut is one induced by a connected component after removing an edge in T.

exception is the case when G is a tree and a trivial greedy algorithm suffices to route H. We prove the theorem by giving a simple (but not so simple) algorithm to compute a routing.

To prove this theorem, we need the following 2-node reduction lemma which is generally known.

Lemma 1. *Let G be a graph and let H be a collection of demands that satisfies the cut condition. Let $G_1, ..., G_k$ be the blocks of G (the 2-node connected components and the cut edges (aka bridges) of G). Let H_i be the collection of nontrivial (i.e., non-loop) demands after contracting each edge $e \in E(G) \setminus E(G_i)$. Then each G_i, H_i satisfies the cut condition. Furthermore, if G (or $G + H$) was outerplanar (or planar), then each G_i (resp. $G_i + H_i$) is outerplanar (resp. planar). Moreover, if each H_i is routable in G_i, then H is routable in G.*

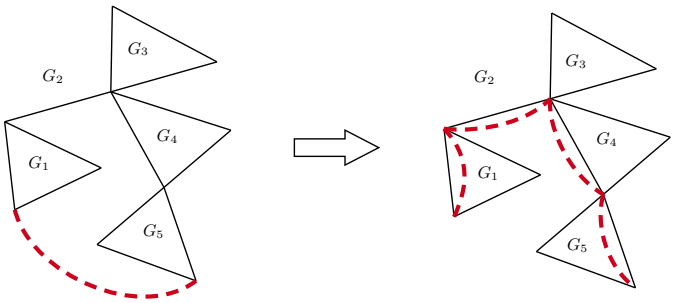

Fig. 1. The new demand edges that replace a demand edge whose terminals belong in different blocks. Solid edges represent edges of G and dashed edges represent demand edges.

Proof. Consider the edge contractions to be done on $G + H$ to obtain $G_i + H_i$. Then, any cut in $G_i + H_i$ was also a cut in $G + H$. Since G, H satisfies the cut condition, then G_i, H_i must also satisfy the cut condition. Furthermore, edge contraction preserves planarity and outerplanarity.

For each $st \in H$ and each G_i, the reduction process produces a request $s_i t_i$ in G_i. If this is not a loop, then s_i, t_i lie in different components of G after deleting the edges of G_i. In this case, we say that *st spawns $s_i t_i$*. Let J be the set of edges spawned by a demand st. It is easy to see that the edges of J form an st path. Hence if each H_i is routable in G_i, we have that H is routable in G. □

Proof (Proof of theorem 2). Without loss of generality, we may assume that the edges of G (resp. H) have unit capacity (resp. demand). Otherwise, we may place $u(e)$ (resp. $d(e)$) parallel copies of such an edge e. In the algorithmic proof, we may also assume that G is 2-node connected. Otherwise, we may apply Lemma 1 and consider each 2-node connected component of G separately. When working

with 2-node connected G, the boundary of its outer face is a simple cycle. So we label the nodes $v_1, ..., v_n$ by the order they appear on this cycle.

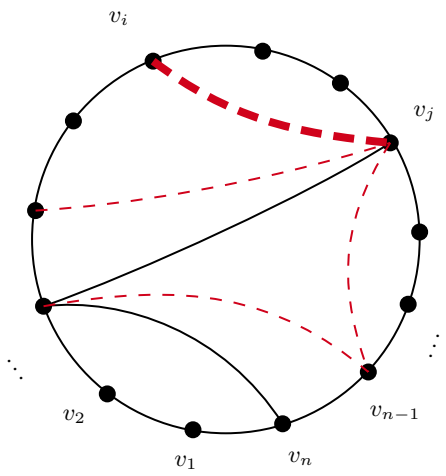

Fig. 2. The solid edges form the outerplanar graph G. The dashed edges are the demand edges. The thick dashed edge is a valid edge to route because there are no terminals v_k with $i < k < j$.

If there are no demand edges, then we are done. Otherwise, since $G + H$ is outerplanar, without loss of generality there exists $i < j$ such that $v_i v_j \in E(H)$ and no v_k is a terminal for $i < k < j$ (Figure 2). Consider the outer face path $P = v_i, v_{i+1}, ..., v_j$. We show that the cut condition is still satisfied after removing both the path P and the demand $v_i v_j$. This represents routing the demand $v_i v_j$ along the path P. Then by induction on the remaining graph $G' + H'$ that is also outerplanar, it will follow that H is routable in G.

Consider a *central cut* $\delta_G(X)$, that is one where $X, V \setminus X$ both induce connected subgraphs. Suppose that v_i and v_j are on opposite sides of the cut. Then, we decrease both $\delta_G(X)$ and $\delta_H(X)$ by 1, so the cut condition holds. Suppose that $v_i, v_j \notin X$, that is v_i and v_j are on the same side of the cut. Then, either $X \subset V(P) \setminus \{v_i, v_j\}$ or $X \cap V(P) = \varnothing$. We are done if $X \cap V(P) = \varnothing$ because $\delta_G(X) \cap E(P) = 0$. Otherwise, $X \subset V(P) \setminus \{v_i, v_j\}$ contains no terminals, so we cannot violate the cut condition. □

We also need the following result from [8].

Theorem 5. *Let T be a tree with integer edge capacities $u(e)$ and k a positive integer. Let H denote a demand graph such that each fundamental cut of H induced by an edge $e \in T$ contains at most $ku(e)$ edges of H. We may then partition H into at most $4k$ edges sets $H_1, \ldots, H_{4k}$ such that each H_i is routable in T.*

2.2 Proof of the Main Theorem

Theorem 1. *There is a polynomial time* 224*-approximation algorithm for the maximum weight* ANF *and* EDP *problems for capacitated outerplanar graphs.*

Proof. We first run the algorithm associated with Theorem 3 to produce an integer-capacitated tree $T, \hat{u}$ which is a 14 cut approximator for G. In addition T is a subtree and it is a conservative approximator for each cut in G. First, we prove that the maximum weight routable in T is not too much smaller than for G (in either the EDP or ANF model). To see this let S be an optimal solution in G, whose value is OPT(G). Clearly S satisfies the cut condition in G and hence by Theorem 3 it satisfies 14 times the cut condition in $T, \hat{u}$. Thus by Theorem 5 there are 56 sets such that $S = \cup_{i=1}^{56} S_i$ and each S_i is routable in T. Hence one of the sets S_i accrues at least $\frac{1}{56}^{th}$ the profit from OPT(G).

Now we use the factor 4 approximation [8] to solve the maximum EDP (equivalent to ANF) problem for $T, \hat{u}$. Let S be a subset of requests which are routable in T and have weight at least $\frac{1}{4}$ OPT(T) $\geq \frac{1}{224}$ OPT(G). Since T is a subtree of G we have that $G+T$ is outerplanar. Since $T, \hat{u}$ is an under-estimator of cuts in G, we have that the edges of T (viewed as requests) satisfies the cut condition in G. Hence by Theorem 2 we may route these single edge requests in G. Hence since S can route in T, we have that S can also route in G, completing the proof. □

3 Single spanning tree cut approximator in Outerplanar Graphs

In this section we first show the existence of a single-tree which is an $O(1)$ cut approximator for an outerplanar graph G. Subsequently we show that there is such a tree with two additional properties. First, its capacity on every cut is at most the capacity in G, and second, all of its weights are integral. These additional properties (integrality and conservativeness) are needed in our application to EDP. The formal statement we prove is as follows.

Theorem 3. *For any connected outerplanar graph $G = (V, E)$ with integer edge capacities $u(e) > 0$, there is a subtree T of G with integer edge weights $\hat{u}(e) \geq 0$ such that*

$$\frac{1}{14} u(\delta_G(X)) \leq \hat{u}(\delta_T(X)) \leq u(\delta_G(X)) \text{ for each proper subset } X \subseteq V.$$

Moreover, the tree may be computer in polynomial time.

In Section 3.1, we show how to view capacity approximators in G as (constrained) distance tree approximators in the planar dual graph. From then on, we look for distance approximators in the dual which correspond to trees in G. In Section 3.2 we prove there exists a single-subtree cut approximator. In Section 3.3 we show how to make this conservative while maintaining integrality of the capacities. In Section 3.4 we prove Theorem 4, which shows that we cannot achieve a single tree bound such as Theorem 3 using the original edge weights.

3.1 Converting flow-sparsifiers in outerplanar graphs to distance-sparsifiers in trees

Let $G = (V, E)$ be an outerplanar graph with capacities $u : E \to \mathbb{R}^+$. Without loss of generality, we can assume that G is 2-node connected, so the boundary of the outer face of G is a cycle that contains each node exactly once. Let G^* be the dual of G; we assign weights to the dual edges in G^* equal to the capacities on the corresponding edges in G. Let G_z be the graph obtained by adding an apex node z to G which is connected to each node of G, that is $V(G_z) = V \cup \{z\}$ and $E(G_z) = E \cup \{(z, v) : v \in V\}$. We may embed z into the outer face of G, so G_z is planar. Let G_z^* denote the planar dual of G_z.

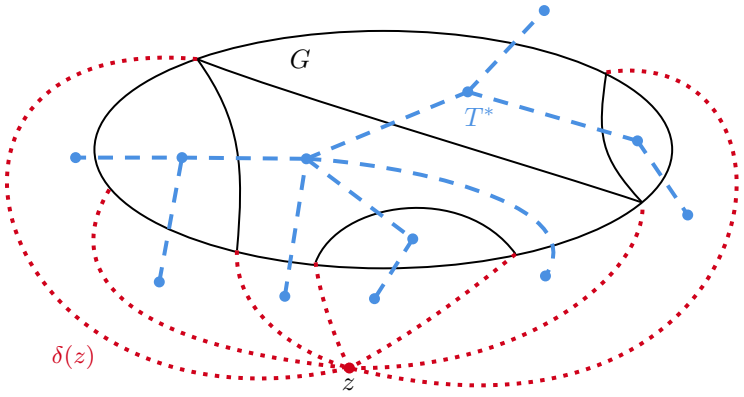

Fig. 3. The solid edges form the outerplanar graph G, and the dotted edges are the edges incident to the apex node z in G_z. The dashed edges form the dual tree T^*.

Note that $\delta(z) = \{(z, v) : v \in V\}$ are the edges of a spanning tree of G_z, so $E(G_z)^* \setminus \delta(z)^*$ are the edges of a spanning tree T^* of G_z^*. Each non-leaf node of T^* corresponds to an inner face of G, and each leaf of T^* corresponds to a face of G_z whose boundary contains the apex node z. Also note that we obtain G^* if we combine all the leaves of T^* into a single node (which would correspond to the outer face of G). We will call T^* the dual tree of the outerplanar graph G (Figure 3).

Let a central cut of G be a cut $\delta(S)$ such that both of its shores S and $V \setminus S$ induce connected subgraphs of G. Hence, the shores of a central cut are subpaths of the outer cycle, so the dual of $\delta(S)$ is a leaf-to-leaf path in T^*. Since the edges of any cut in a connected graph is a disjoint union of central cuts, it suffices to only consider central cuts.

We want to find a strictly embedded cut-sparsifier $T = (V, F, u^*)$ of G (i.e. a spanning tree T of G with edges weights u^*) such that for any nonempty $X \subsetneq V$,

we have

$$\alpha u(\delta_G(X)) \leq u^*(\delta_T(X)) \leq \beta u(\delta_G(X)). \tag{1}$$

In the above inequality, we can replace $u^*(\delta_T(X))$ with $u^*(\delta_G(X))$ if we set $u^*(e) = 0$ for each edge $e \notin E(T)$. In the dual tree (of G), $\delta_G(X)^*$ is a leaf-to-leaf path for any central cut $\delta(X)$, so inequality (1) is equivalent to

$$\alpha u(P) \leq u^*(P) \leq \beta u(P) \tag{2}$$

for any leaf-to-leaf path P in T^*.

Finally, we give a sufficient property on the weights u^* assigned to the edges such that all edges of positive weight are in the spanning tree of G. Recall that the dual of the edges not in the spanning tree of G would form a spanning tree of G^*. Since we assign weight 0 to edges not in the spanning tree of G, it is sufficient for the 0 weight edges to form a connected spanning subgraph of G^*. Since G^* is obtained by combining the leaves of T^* into a single node, it suffices for each node $v \in V(T^*)$ to have a 0 weight path from v to a leaf of T^*.

3.2 An algorithm to build a distance-sparsifier of a tree

In this section, we present an algorithm to obtain a distance-sparsifier of a tree. In particular, this allows us to obtain a cut-approximator of an outerplanar graph from a distance-sparsifier of its dual tree.

Let $T = (V, E, u)$ be a weighted tree where $u : E \to \mathbb{R}^+$ is the length function on T. Let $L \subset V$ be the leaves of T. We assign non-negative weights u^* to the edges of T. Let d be the shortest path metric induced by the original weights u, and let d^* be the shortest path metric induced by the new weights u^*. We want the following two conditions to hold:

1. there exists a 0 weight path from each $v \in V$ to a leaf of T.
2. for any two leaves $x, y \in L$, we have

$$\frac{1}{4} d(x, y) \leq d^*(x, y) \leq 2d(x, y). \tag{3}$$

We define u^* recursively as follows. Let r be a non-leaf node of T (we are done if no such nodes exist), and consider T to be rooted at r. For $v \in V$, let $T(v)$ denote the subtree rooted at v, and let $h(v)$ denote the *height* of v, defined by $h(v) = \min\{d(v, x) : x \in L \cap T(v)\}$. Now, let $r_1, ..., r_k$ be the points in T that are at distance exactly $h(r)/2$ from r. Without loss of generality, suppose that each r_i is a node (otherwise we can subdivide the edge to get a node), and order the r_i's by increasing $h(r_i)$, that is $h(r_{i-1}) \leq h(r_i)$ for each $i = 2, ..., k$. Furthermore, suppose that we have already assigned weights to the edges in each subtree $T(r_i)$ using this algorithm, so it remains to assign weights to the edges not in any of these subtrees. We assign a weight of $h(r_i)$ to the first edge on the path from r_i to r for each $i = 2, ..., k$, and weight 0 to all other edges (Figure 4). In particular, all edges on the path from r_1 to r receive weight 0.

The algorithm terminates on components with at most one vertex in L. Let $w \in L$ be a leaf closest to r, $d(r, w) = h(r)$, and let ℓ be the length of the edge incident to w. As the length of the longest path from the root to w is halved at each recursive step, w will be isolated after at most $log_2 \frac{h(r)}{\ell}$ recursive steps.

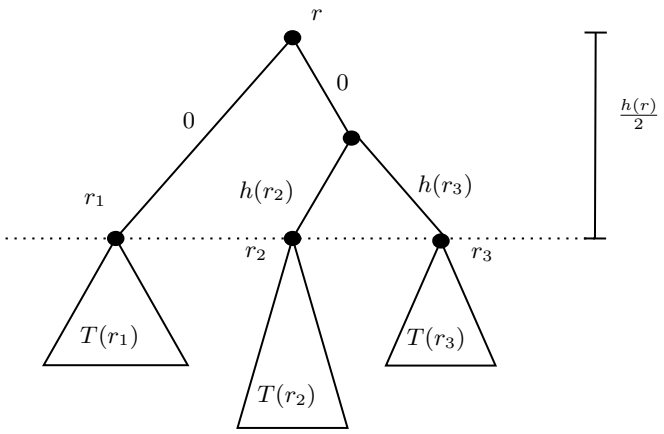

Fig. 4. The algorithm assigns weights to the edges above $r_1, ..., r_k$, and is run recursively on the subtrees $T(r_1), ..., T(r_k)$.

Since we assign 0 weight to edges on the $r_1 r$ path, Condition 1 is satisfied for all nodes above the r_i's in the tree by construction. It remains to prove Condition 2. We use the following upper and lower bounds. For each leaf $x \in L$,

$$d^*(x, r) \leq 2d(x, r) - h(r), \tag{4}$$
$$d^*(x, r) \geq d(x, r) - h(r). \tag{5}$$

We prove the upper bound in (4) by induction. We are done if T only has 0 weight edges, and the cases that cause the algorithm to terminate will only have 0 weight edges. For the induction, we consider two separate cases depending on whether $x \in T(r_1)$.

Case 1: $x \in T(r_1)$.

$$\begin{aligned}
d^*(x, r) &= d^*(x, r_1) + d^*(r_1, r) && (r_1 \text{ is between } x \text{ and } r)\\
&= d^*(x, r_1) && (\text{by definition of } u^*)\\
&\leq 2d(x, r_1) - h(r_1) && (\text{by induction})\\
&= 2d(x, r) - 2d(r, r_1) - h(r_1) && (r_1 \text{ is between } x \text{ and } r)\\
&= 2d(x, r) - \frac{3}{2}h(r) && (h(r_1) = h(r)/2 \text{ by definition of } r_1)\\
&\leq 2d(x, r) - h(r)
\end{aligned}$$

Case 2: $x \in T(r_i)$ for some $i \neq 1$.

$$\begin{aligned}
d^*(x,r) &= d^*(x,r_i) + d^*(r_i,r) && (r_i \text{ is between } x \text{ and } r)\\
&= d^*(x,r_i) + h(r_i) && (\text{by definition of } u^*)\\
&\leq 2d(x,r_i) - h(r_i) + h(r_i) && (\text{by induction})\\
&= 2d(x,r) - 2d(r_i,r) && (r_i \text{ is between } x \text{ and } r)\\
&= 2d(x,r) - h(r) && (d(r_i,r) = h(r)/2 \text{ by definition of } r_i)
\end{aligned}$$

This proves inequality (4).

We prove the lower bound in (5) similarly.

Case 1: $x \in T(r_1)$.

$$\begin{aligned}
d^*(x,r) &= d^*(x,r_1) + d^*(r_1,r) && (r_1 \text{ is between } x \text{ and } r)\\
&= d^*(x,r_1) && (\text{by definition of } u^*)\\
&\geq d(x,r_1) - h(r_1) && (\text{by induction})\\
&= d(x,r) - d(r,r_1) - h(r_1) && (r_1 \text{ is between } x \text{ and } r)\\
&= d(x,r) - h(r) && (\text{by definition of } r_1)
\end{aligned}$$

Case 2: $x \in T(r_i)$ for some $i \neq 1$.

$$\begin{aligned}
d^*(x,r) &= d^*(x,r_i) + d^*(r_i,r) && (r_i \text{ is between } x \text{ and } r)\\
&= d^*(x,r_i) + h(r_i) && (\text{by definition of } u^*)\\
&\geq d(x,r_i) - h(r_i) + h(r_i) && (\text{by induction})\\
&= d(x,r) - d(r_i,r) && (r_i \text{ is between } x \text{ and } r)\\
&= d(x,r) - h(r)/2 && (d(r_i,r) = h(r)/2 \text{ by definition of } r_i)\\
&\geq d(x,r) - h(r)
\end{aligned}$$

This proves inequality (5)

Finally, we prove Condition 2, that is inequality (3), by induction. Let $x, y \in L$ be two leaves of T. Suppose that $x \in T(r_i)$ and $y \in T(r_j)$. By induction, we may assume that $i \neq j$, so without loss of generality, suppose that $i < j$.

We prove the upper bound.

$$\begin{aligned}
d^*(x,y) &= d^*(x,r_i) + d^*(r_i,r_j) + d^*(r_j,y)\\
&\leq 2d(x,r_i) - h(r_i) + 2d(y,r_j) - h(r_j) + d^*(r_i,r_j) && (\text{by (4)})\\
&\leq 2d(x,r_i) - h(r_i) + 2d(y,r_j) - h(r_j) + h(r_i) + h(r_j) && (\text{by definition of } u^*)\\
&= 2d(x,r_i) + 2d(y,r_j)\\
&\leq 2d(x,y)
\end{aligned}$$

We prove the lower bound.

$$\begin{aligned}
d(x,y) &= d(x,r_i) + d(r_i,r_j) + d(r_j,y) \\
&\le d(x,r_i) + d(r_j,y) + h(r_i) + h(r_j) \\
&\qquad \text{(because } d(r,r_i) = h(r)/2 \le h(r_i) \text{ for all } i \in [k]) \\
&\le 2d(x,r_i) + 2d(r_j,y) && \text{(by definition of } h) \\
&\le 2d^*(x,r_i) + 2h(r_i) + 2d^*(y,r_j) + 2h(r_j) && \text{(by (5))} \\
&= 2d^*(x,y) - 2d^*(r_i,r_j) + 2h(r_i) + 2h(r_j).
\end{aligned}$$

Now we finish the proof of the lower bound by considering two cases.

Case 1: $i = 1$, that is x is in the first subtree.

$$\begin{aligned}
d(x,y) &\le 2d^*(x,y) - 2d^*(r_1,r_j) + 2h(r_1) + 2h(r_j) \\
&= 2d^*(x,y) - 2h(r_j) + 2h(r_1) + 2h(r_j) && \text{(by definition of } u^*) \\
&\le 2d^*(x,y) + 2h(r_1) \\
&\le 4d^*(x,y)
\end{aligned}$$

Case 2: $i > 1$, that is neither x nor y is in the first subtree.

$$\begin{aligned}
d(x,y) &\le 2d^*(x,y) - 2d^*(r_i,r_j) + 2h(r_i) + 2h(r_j) \\
&= 2d^*(x,y) - 2h(r_i) - 2h(r_j) + 2h(r_i) + 2h(r_j) && \text{(by definition of } u^*) \\
&= 2d^*(x,y)
\end{aligned}$$

This completes the proof of Condition 2.

3.3 Converting to Integer Weight Cut-Conservative Approximators

Section 3.2 described an algorithm to get

$$\frac{1}{4}d(x,y) \le d^*(x,y) \le 2d(x,y) \text{ for leaves } x,y \in L.$$

However, the assigned weights were not necessarily integers. In this section, we modify the algorithm to assign integer weights, and achieve the following bound:

$$\frac{1}{14}d(x,y) \le d^*(x,y) \le d(x,y) \text{ for leaves } x,y \in L. \tag{6}$$

The algorithm will be the same as before except for two changes. First, we choose $r_1, ..., r_k$ to be the points that are exactly distance $\lceil h(r)/2 \rceil$ away from r. Second, we assign a weight of $\max\{\lfloor h(r_i)/2 \rfloor, 1\}$ to the first edge on the path from r_i to r for each $i = 2, ..., k$ (Figure 5). Note that this algorithm also terminates because the length of the longest path from the root to a leaf decreases by at least half the length of the shortest edge incident to a leaf in each iteration.

Since we assign 0 weight to edges on the path from r_1 to r, there exists a 0 weight path from each node to a leaf. It remains to prove that the assigned weights satisfy the bound.

To prove the upper bound $d^*(x,y) \le d(x,y)$, we need the following lemma.

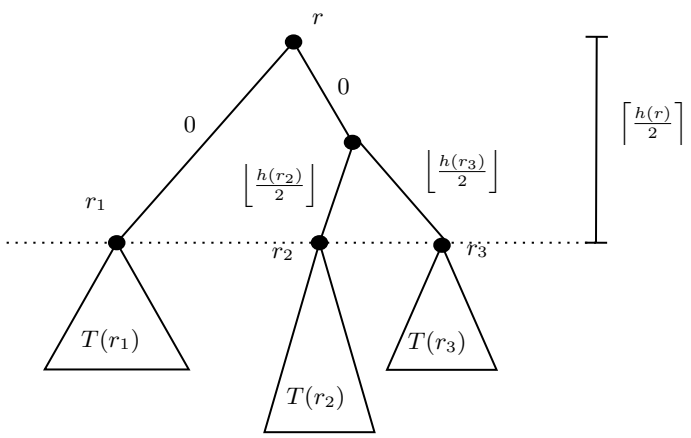

Fig. 5. The algorithm assigns integer weights to the edges above $r_1, ..., r_k$, and is run recursively on the subtrees $T(r_1), ..., T(r_k)$.

Lemma 2. *For each leaf $x \in L$,*

$$2d^*(x,r) \leq 2d(x,r) - h(r) + 2. \tag{7}$$

Furthermore, if $h(r) \leq 1$, we have

$$d^*(x,r) \leq d(x,r). \tag{8}$$

Proof. We prove the upper bounds in (7) and (8) simultaneously by induction. We are done if T is a single node, otherwise we consider a few cases depending on whether $x \in T(r_1)$ and whether $h(r_i) \geq 2$.

Case 1 (inequality (7)): $x \in T(r_1)$.

$$\begin{aligned}
2d^*(x,r) &= 2d^*(x,r_1) + 2d^*(r_1,r) && (r_1 \text{ is between } x \text{ and } r)\\
&= 2d^*(x,r_1) && (\text{by definition of } u^*)\\
&\leq 2d(x,r_1) - h(r_1) + 2 && (\text{by induction with (7)})\\
&= 2d(x,r) - 2d(r,r_1) - h(r_1) + 2 && (r_1 \text{ is between } x \text{ and } r)\\
&= 2d(x,r) - (\lceil h(r)/2 \rceil + h(r)) + 2 && (\text{since } d(r_1,r) + h(r_1) = h(r))\\
&\leq 2d(x,r) - h(r) + 2
\end{aligned}$$

Case 2 (inequality (7)): $x \in T(r_i)$ for some $i \neq 1$, and $h(r_i) \geq 2$. Note that this means $\lfloor h(r_i)/2 \rfloor \geq 1$.

$$\begin{aligned}
2d^*(x,r) &= 2d^*(x,r_i) + 2d^*(r_i,r) && (r_i \text{ is between } x \text{ and } r)\\
&= 2d^*(x,r_i) + 2\lfloor h(r_i)/2 \rfloor && (\text{by definition of } u^*)\\
&\leq 2d(x,r_i) - h(r_i) + 2 + h(r_i) && (\text{by induction with (7)})\\
&= 2d(x,r) - 2d(r_i,r) + 2 && (r_i \text{ is between } x \text{ and } r)\\
&= 2d(x,r) - 2\lceil h(r)/2 \rceil + 2 && (\text{since } d(r_i,r) = \lceil h(r)/2 \rceil)\\
&\leq 2d(x,r) - h(r) + 2
\end{aligned}$$

Case 3 (inequality (7)): $x \in T(r_i)$ and for some $i \neq 1$, $h(r_i) \leq 1$.

$$\begin{aligned}
2d^*(x,r) &= 2d^*(x,r_i) + 2d^*(r_i,r) && (r_i \text{ is between } x \text{ and } r)\\
&= 2d^*(x,r_i) + 2 && (\text{by definition of } u^*)\\
&\leq 2d(x,r_i) + 2 && (\text{by induction with (8)})\\
&= 2d(x,r) - 2d(r_i,r) + 2 && (r_i \text{ is between } x \text{ and } r)\\
&= 2d(x,r) - 2\lceil h(r)/2 \rceil + 2 && (d(r_i,r) = \lceil h(r)/2 \rceil \text{ by definition of } r_i)\\
&\leq 2d(x,r) - h(r) + 2
\end{aligned}$$

For the remaining cases, we assume that $h(r) = 1$, so $d(r,r_i) = 1$ for each i.

Case 4 (inequality (8)): $x \in T(r_1)$. Since $h(r) = 1$, we necessarily have $x = r_1$ and $d(x,r) = 1$, so

$$d^*(x,r) = 0 \leq 1 = d(x,r).$$

Case 5 (inequality (8)): $x \in T(r_i)$ for some $i \neq 1$ and $h(r_i) \geq 2$.

$$\begin{aligned}
2d^*(x,r) &= 2d^*(x,r_i) + 2d^*(r_i,r) && (r_i \text{ is between } x \text{ and } r)\\
&= 2d^*(x,r_i) + 2\lfloor h(r_i)/2 \rfloor && (\text{by definition of } u^*)\\
&\leq 2d(x,r_i) - h(r_i) + 2 + h(r_i) && (\text{by induction with (7)})\\
&= 2d(x,r) && (\text{because } d(r,r_i) = 1)
\end{aligned}$$

Case 6 (inequality (8)): $x \in T(r_i)$ and for some $i \neq 1$ $h(r_i) \leq 1$.

$$\begin{aligned}
2d^*(x,r) &= 2d^*(x,r_i) + 2d^*(r_i,r) && (r_i \text{ is between } x \text{ and } r)\\
&= 2d^*(x,r_i) + 2 && (\text{by definition of } u^*)\\
&\leq 2d(x,r_i) + 2 && (\text{by induction with (8)})\\
&= 2d(x,r) && (\text{because } d(r,r_i) = 1)
\end{aligned}$$

□

To prove the lower bound $d(x,y) \leq 14d^*(x,y)$, we need the following lemma.

Lemma 3. *For each leaf x,*

$$d(x,r) \leq 3d^*(x,r) + h(r). \tag{9}$$

Proof. We prove the lower bound in (9) by induction.

Case 1: $x \in T(r_1)$.

$$\begin{aligned}
3d^*(x,r) &= 3d^*(x,r_1) + 3d^*(r_1,r) && (r_1 \text{ is between } x \text{ and } r)\\
&= 3d^*(x,r_1) && (\text{by definition of } u^*)\\
&\geq d(x,r_1) - h(r_1) && (\text{by induction})\\
&= d(x,r) - d(r,r_1) - h(r_1) && (r_1 \text{ is between } x \text{ and } r)\\
&= d(x,r) - h(r) && (\text{by definition of } r_1)
\end{aligned}$$

Rearranging gives the desired inequality.

Case 2: $x \in T(r_i)$ for some $i \neq 1$.

$$\begin{aligned}
3d^*(x,r) &= 3d^*(x,r_i) + 3d^*(r_i,r) && (r_i \text{ is between } x \text{ and } r)\\
&\geq d(x,r_i) - h(r_i) + 3d^*(r_i,r) && (\text{by induction})\\
&= d(x,r_i) - h(r_i) + 3\max\{1, \lfloor h(r_i)/2 \rfloor\} && (\text{by definition of } u^*)\\
&\geq d(x,r_i) - h(r_i) + 1 + 2\lfloor h(r_i)/2 \rfloor\\
&\geq d(x,r_i)\\
&= d(x,r) - d(r_i,r) && (r_i \text{ is between } x \text{ and } r)\\
&\geq d(x,r) - h(r) && (\text{since } d(r_i,r) \leq h(r))
\end{aligned}$$

Rearranging gives the desired inequality. □

Finally, we prove the bounds in (6) by induction. Let $x, y \in L$ be two leaves of T. Suppose that $x \in T(r_i)$ and $y \in T(r_j)$. By induction, we may assume that $i \neq j$ (otherwise we may consider the subtree rooted at r_i since the algorithm is recursive). Without loss of generality, suppose that $i < j$.

We prove the upper bound. Let c be the lowest common ancestor of x and y. Since $r_i \neq r_j$, the node c is also an ancestor of r_i and r_j, so $d(r_i,c) \geq 1$ and $d(r_j,c) \geq 1$. First we show that $d^*(x,c) \leq d(x,c)$ by considering two cases.

Case 1: $h(r_i) \geq 2$.

$$\begin{aligned}
2d^*(x,c) &= 2d^*(x,r_i) + 2d^*(r_i,c) && (r_i \text{ is between } x \text{ and } c)\\
&\leq 2d(x,r_i) - h(r_i) + 2 + 2d^*(r_i,c) && (\text{by upper bound (7)})\\
&\leq 2d(x,r_i) - h(r_i) + 2 + 2\lfloor h(r_i)/2 \rfloor && (\text{by definition of } u^*)\\
&\leq 2d(x,r_i) + 2 && (\text{since } 2\lfloor h(r_i)/2 \rfloor \leq h(r_i))\\
&\leq 2d(x,r_i) + 2d(r_i,c) && (\text{since } d(r_i,c) \geq 1)\\
&= 2d(x,c)
\end{aligned}$$

Case 2: $h(r_i) \leq 1$.

$$\begin{aligned}
2d^*(x,c) &= 2d^*(x,r_i) + 2d^*(r_i,c) && (r_i \text{ is between } x \text{ and } c)\\
&\leq 2d(x,r_i) + 2d^*(r_i,c) && (\text{by upper bound (8)})\\
&= 2d(x,r_i) + 2 && (\text{by definition of } u^*)\\
&\leq 2d(x,c) && (\text{since } d(r_i,c) \geq 1)
\end{aligned}$$

The proof that $d^*(y,c) \leq 2d(y,c)$ is the same. We conclude that

$$d^*(x,y) = d^*(x,c) + d^*(y,c) \leq d(x,c) + d(y,c) = d(x,y),$$

and the upper bound is proved.

We prove the lower bound. First note that $i < j$ means

$$\begin{aligned}
14d^*(r_i,r_j) &\geq 14\max\{1, \lfloor h(r_j)/2 \rfloor)\}\\
&\geq 6 \cdot 1 + 8 \cdot \lfloor h(r_j)/2 \rfloor\\
&\geq 2 + 4h(r_j).
\end{aligned}$$

Then, we can compute

$$\begin{aligned}
d(x,y) &= d(x,r_i) + d(r_i,r_j) + d(r_j,y) \\
&\le d(x,r_i) + d(y,r_j) + d(r,r_i) + d(r,r_j) \\
&\le d(x,r_i) + d(y,r_j) + h(r_i) + h(r_j) + 2 && \text{(since } d(r,r_i) \le h(r_i)+1\text{)} \\
&\le 3d^*(x,r_i) + 3d^*(y,r_j) + 2h(r_i) + 2h(r_j) + 2 && \text{(by (9))} \\
&\le 14d^*(x,r_i) + 14d^*(y,r_j) + 4h(r_j) + 2 \\
&\le 14d^*(x,y) - 14d^*(r_i,r_j) + 4h(r_j) + 2 \\
&\le 14d^*(x,y) - 2 - 4h(r_j) + 4h(r_j) + 2 \\
&\le 14d^*(x,y).
\end{aligned}$$

This completes the proofs of the inequalities (6). □

3.4 Lower bound on Congestion in the Exact Weight Model

In this section we prove Theorem 4.

Proof. Let G be an undirected unit capacity graph with 3×2^n nodes defined as follows. Label the nodes of G with the integers 0 to $3 \times 2^n - 1$, and arrange the nodes in a circle so that node $i+1$ comes after node i. First add the three edges $(0, 2^n)$, $(2^n, 2^{n+1})$, and $(2^{n+1}, 0)$. Then, recursively for each edge (u, v) that is not between consecutive nodes on the circle, add the two edges $(u, (u+v)/2)$ and $((u+v)/2, v)$ (Figure 6). We show that any spanning tree of G has $\Omega(n)$ congestion.

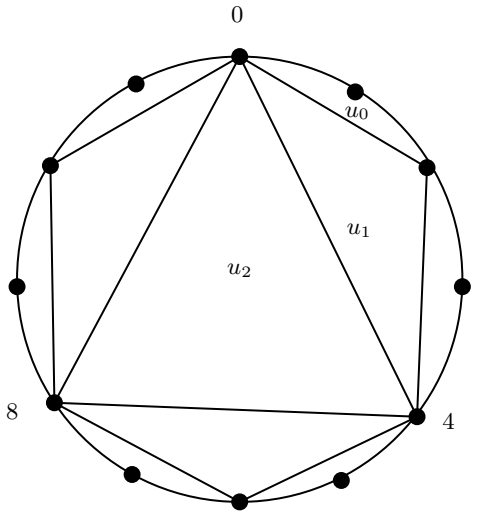

Fig. 6. The congestion lower bound graph for $n = 2$.

Since G is planar and 2-connected, by Andersen & Feige [3], we know that every spanning tree of G with congestion ρ gives a spanning tree of the planar dual of G with stretch at most $\rho + 1$. Hence, it suffices to show that every spanning tree of the planar dual of G has $\Omega(n)$ stretch.

Let $\bar{G}$ be the planar dual of G and let $v \in V(\bar{G})$ be the node that corresponds to the outer face of G. Let $u_k \in V(\bar{G})$ be the node that corresponds to the face of G bounded by the triangle $0, 2^k, 2^{k+1}$ for $k = 0, ..., n$. We show that the distance between v and u_n is $n + 1$. By symmetry, any path from v to u_n has the same length, so we just need to find the length of one such path. Note that each u_k is adjacent to u_{k-1} for $k = 1, ..., n$, and u_0 is adjacent to v, so $v, u_0, u_1, ..., u_n$ is a path from v to u_n with length $n + 1$. Also note that there are at least two node disjoint paths from v to u_n, so the shortest cycle containing both v and u_n has length $2n + 2$. Let C denote this cycle. By [26], any dominating tree of C has stretch $\Omega(|V(C)|) = \Omega(n)$. In particular, any spanning tree of $\bar{G}$ is a dominating tree of C, so any spanning tree of $\bar{G}$ has stretch $\Omega(n)$.

We conclude that any spanning tree of G has congestion $\Omega(n) = \Omega(\log |V(G)|)$. □

4 Conclusions

The technique of finding a single-tree constant-factor cut approximator (for a global constant) appears to hit a limit at outerplanar graphs. It would be interesting to find a graph parameter k which ensures a single-tree $O(f(k))$ cut approximator.

The authors thank Nick Harvey for his valuable feedback on this article. The authors are also grateful for excellent feedback from anonymous reviewers from both Math Programming Series B, and the conference IPCO. The last two authors are grateful for the support of an NSERC Discovery Grant which made this work possible. The first author is partially supported by ANR project DISTANCIA (ANR-17-CE40-0015).

References

1. Ittai Abraham, Yair Bartal, and Ofer Neiman. Nearly tight low stretch spanning trees. In *FOCS*, pages 781–790, 2008.
2. Noga Alon, Richard M Karp, David Peleg, and Douglas West. A graph-theoretic game and its application to the k-server problem. *SIAM Journal on Computing*, 24(1):78–100, 1995.
3. Reid Andersen and Uriel Feige. Interchanging distance and capacity in probabilistic mappings. *arXiv preprint arXiv:0907.3631*, 2009.
4. C. Chekuri, S. Khanna, and F.B. Shepherd. Edge-disjoint paths in planar graphs with constant congestion. *SIAM Journal on Computing*, 39:281–301, 2009.
5. Chandra Chekuri and Alina Ene. Poly-logarithmic approximation for maximum node disjoint paths with constant congestion. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 326–341. SIAM, 2013.

6. Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. The all-or-nothing multicommodity flow problem. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 156–165, New York, NY, USA, 2004. ACM.
7. Chandra Chekuri, Sanjeev Khanna, and F Bruce Shepherd. The all-or-nothing multicommodity flow problem. *SIAM Journal on Computing*, 42(4):1467–1493, 2013.
8. Chandra Chekuri, Marcelo Mydlarz, and F Bruce Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Transactions on Algorithms (TALG)*, 3(3):27–es, 2007.
9. Chandra Chekuri, Guyslain Naves, and F Bruce Shepherd. Maximum edge-disjoint paths in k-sums of graphs. In *International Colloquium on Automata, Languages, and Programming*, pages 328–339. Springer, 2013.
10. J. Chuzhoy and S. Li. A polylogarithimic approximation algorithm for edge-disjoint paths with congestion 2. *arXiv preprint arXiv:1208.1272*, 2012.
11. J. Chuzhoy and S. Li. A polylogarithimic approximation algorithm for edge-disjoint paths with congestion 2. In *Proc. of IEEE FOCS*, 2012.
12. Julia Chuzhoy, David HK Kim, and Rachit Nimavat. New hardness results for routing on disjoint paths. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 86–99, 2017.
13. Michael Elkin, Yuval Emek, Daniel A Spielman, and Shang-Hua Teng. Lower-stretch spanning trees. *SIAM Journal on Computing*, 38(2):608–628, 2008.
14. Matthias Englert, Anupam Gupta, Robert Krauthgamer, Harald Racke, Inbal Talgam-Cohen, and Kunal Talwar. Vertex sparsifiers: New results from old techniques. *SIAM Journal on Computing*, 43(4):1239–1262, 2014.
15. András Frank. Edge-disjoint paths in planar graphs. *Journal of Combinatorial Theory, Series B*, 39(2):164–178, 1985.
16. Naveen Garg, Nikhil Kumar, and András Sebő. Integer plane multiflow maximisation: Flow-cut gap and one-quarter-approximation. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 144–157. Springer, 2020.
17. Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.
18. A. Gupta, I. Newman, Y. Rabinovich, and A. Sinclair. Cuts, trees and ℓ_1-embeddings of graphs. *Combinatorica*, 24(2):233–269, 2004.
19. Anupam Gupta. Steiner points in tree metrics don't (really) help. In *SODA*, volume 1, pages 220–227, 2001.
20. 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. *Journal of Computer and System Sciences*, 67(3):473–496, 2003.
21. Chris Harrelson, Kirsten Hildrum, and Satish Rao. A polynomial-time tree decomposition to minimize congestion. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, pages 34–43, 2003.
22. Chien-Chung Huang, Mathieu Mari, Claire Mathieu, Kevin Schewior, and Jens Vygen. An approximation algorithm for fully planar edge-disjoint paths. *arXiv preprint arXiv:2001.01715*, 2020.
23. Ken-ichi Kawarabayashi and Yusuke Kobayashi. All-or-nothing multicommodity flow problem with bounded fractionality in planar graphs. *SIAM Journal on Computing*, 47(4):1483–1504, 2018.

24. J. Kleinberg and E. Tardos. Approximations for the disjoint paths problem in high-diameter planar networks. *Journal of Computers and System Sciences*, 57(1):61–73, 1998.
25. Haruko Okamura and P. D. Seymour. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory, Series B*, 31(1):75–81, 1981/8.
26. Yuri Rabinovich and Ran Raz. Lower bounds on the distortion of embedding finite metric spaces in graphs. *Discrete & Computational Geometry*, 19(1):79–94, 1998.
27. H. Räcke. Minimizing congestion in general networks. In *Proc. of IEEE FOCS*, pages 43–52, 2002.
28. Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *STOC*, pages 255–264, 2008.
29. Harald Räcke and Chintan Shah. Improved guarantees for tree cut sparsifiers. In *European Symposium on Algorithms*, pages 774–785. Springer, 2014.
30. L. Seguin-Charbonneau and F.B. Shepherd. Maximum edge-disjoint paths in planar graphs with congestion 2. *Math Programming*, 2020.
31. Paul D Seymour. Matroids and multicommodity flows. *European Journal of Combinatorics*, 2(3):257–290, 1981.
32. Anastasios Sidiropoulos. Private communication, 2014.