Edge-Disjoint Paths in Planar Graphs with Constant Congestion

Chandra Chekuri
Lucent Bell Labs
600 Mountain Avenue
Murray Hill, NJ 07974
chekuri@research.bell-labs.com

Sanjeev Khanna
Dept. of Comp. & Inf. Sci.
University of Pennsylvania
Philadelphia, PA 19104
sanjeev@cis.upenn.edu

F. Bruce Shepherd
Lucent Bell Labs
600 Mountain Avenue
Murray Hill, NJ 07974
bshep@research.bell-labs.com

ABSTRACT

We study the maximum edge-disjoint paths problem in undirected planar graphs: given a graph G and node pairs s_1t_1, s_2t_2, $\ldots$, s_kt_k, the goal is to maximize the number of pairs that can be connected (routed) by edge-disjoint paths. The natural multicommodity flow relaxation has an $\Omega(\sqrt{n})$ integrality gap. Motivated by this, we consider solutions with small constant *congestion* $c > 1$; that is, solutions in which up to c paths are allowed to use an edge (alternatively, each edge has a capacity of c). In previous work we obtained an $O(\log n)$ approximation with congestion 2 via the flow relaxation. This was based on a method of decomposing into *well-linked* subproblems.

In this paper we obtain an $O(1)$ approximation with congestion 4. To obtain this improvement we develop an alternative decomposition that is specific to planar graphs. The decomposition produces instances that we call *Okamura-Seymour* (OS) instances. These have the property that all terminals lie on a single face. Another ingredient we develop is a constant factor approximation for the all-or-nothing flow problem on OS instances via the flow relaxation.

We also study limitations on the approximation that can be achieved by a well-linked decomposition. For general graphs we show a lower bound of $\Omega(\log n)$. For planar graphs we describe instances that suggest a super-constant lower bound.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Non-numerical Algorithms and Problems

General Terms

Algorithms, Theory.

Keywords

Edge-disjoint Paths, Planar Graphs, Multicommodity Flow.

STOC'06, May21–23, 2006, Seattle, Washington, USA.

1. INTRODUCTION

1.1 Edge-Disjoint Paths, Cut Condition, and Multicommodity Flow

In this paper we study the *edge-disjoint path* problem (EDP) in undirected graphs. We are given a supply graph $G = (V, E)$ and a demand graph $H = (V, F)$. We sometimes write the demand set F as a set of node pairs $\mathcal{T} = \{s_1t_1, s_2t_2, \ldots, s_kt_k\}$. The objective is to connect the pairs via edge-disjoint paths. If such a collection exists, the demands are said to be *routable*. In the maximum edge-disjoint path problem (MEDP), the objective is to find a maximum routable subset of F (or $\mathcal{T}$). More generally, we sometimes allow each demand to have a specified demand amount d_i, and supply edges to have capacities; we also may relax the notion of routability to suit our purposes. For instance, we may allow the demand to be fractionally routed (*multicommodity flow*) or we may require that the whole demand be routed on a single path (*unsplittable flow*).

The problem of determining whether a set of demands is routable is one of the classical NP-hard problems in combinatorial optimization [9, 14]. There are two natural necessary conditions that are employed in tackling this feasibility problem. The first is the *cut condition*: for every proper subset $S \subseteq V$, $|\delta_G(S)| \geq |\delta_H(S)|$. If demand edges have values, then the right-hand side becomes $\sum_{f \in \delta_H(S)} d_f$, and if edges have capacities, $\delta_G(S)$ is altered accordingly. Here we use $\delta_G(S)$ to denote the set of edges in a graph G with exactly one end point in S. The second necessary condition is the existence of a *multicommodity flow* (MCF) for the demand graph H. That is, an assignment of nonnegative flow $f(P)$ to paths P that obey the following capacity and demand constraints: (1) $\sum_{P: e \in P} f(P) \leq u_e$ and (2) $\sum_{P \in \mathcal{P}_i} f(P) = 1$. We let $\mathcal{P}_i$ denote the paths joining s_i and t_i in G and let $\mathcal{P} = \cup_i \mathcal{P}_i$. In (1) u_e denotes the capacity of edge e.

Obviously if F is routable, then it has a multicommodity flow, and the existence of a flow implies that the cut condition holds. However, there are instances where the cut condition holds but for which there is no flow, and instances where a flow exists but for which F is not routable. Not surprisingly, research has tended to bifurcate into two categories. One is to examine special cases where the cut condition (or more general conditions such as the existence of a flow) guarantees solvability of the disjoint path problem. A second direction is to quantify the degree to which routability (fractional or integral) diverges from the cut condition.

In the first category, there are two archetypal examples that play a central role in the current study. One is the max-flow-min-cut theorem for single-source multicommodity flow. Here we have a source node s and terminals $t_1, t_2, \ldots, t_k$ (possibly also with associated integer demands d_i). Results of [10, 23] then show that the cut condition is sufficient for the simultaneous routability of the demands. The second is the Okamura-Seymour Theorem which states that the cut condition is sufficient for solvability of multicommodity flow for any instance where G is planar and all demand edges have both ends on a single face of G, in particular on the outer face.

In the second category, the notion of *concurrency* plays a rôle in measuring so-called *flow-cut gaps*. Given an instance of EDP or MCF that satisfies the cut condition, one asks for the largest value $\lambda > 0$ such that there exists a multicommodity flow that simultaneously routes λd_f for each demand edge f. The inverse of the worst case value of λ over all all instances in G is the flow-cut gap for G. In [21], a first positive breakthrough was given in bounding the flow-cut gap for general undirected graphs G; a bound of $O(\log n)$ was established for uniform instances of MCF. For the non-uniform case a similar bound was shown in [3, 22]. In planar graphs, for uniform multicommodity flow, the flow-cut gap was shown to be $O(1)$ [15], while for non-uniform multicommodity flow the current best upper bound on the gap is $O(\sqrt{\log n})$ [26].

1.2 Maximum Edge-Disjoint Paths Problem

In tackling the throughput maximization problem MEDP, one has a natural LP relaxation. Assuming the unweighted version, we seek to maximize $\sum_P f(P)$ subject to the constraints (1) and (2) above, where we relax the latter to inequality. If OPT is the optimal value of the LP, then we seek to find a feasible integral solution that guarantees a large fraction of OPT.

Unfortunately, the integrality gap for this LP is $\Omega(\sqrt{n})$, even in the case of planar graphs [13]. This bound is tight even in general graphs [8]. In contrast, known inapproximability bounds are exponentially smaller: only recently a first super-constant lower bound was given in [1]: unless $NP \subseteq ZPTIME(n^{\text{polylog}(n)})$ there is no $O(\log^{\frac{1}{3}-\epsilon} n)$ approximation for MEDP (this hardness factor is improved to $O(\log^{\frac{1}{2}-\epsilon} n)$ in [2]).

In the face of these negative results, it is natural to examine relaxations of the problem. MEDP's difficulty stems from two seemingly distinct aspects: one arising from the gap between fractional and integral flows, and the other stems from the selection of which subset of demands to try to route. When G itself is a tree, the first complication disappears and one may focus on the issues involved in choosing a most profitable routable subset of demands. Even on trees this problem is APX-hard [13]. A 2 approximation for the cardinality case was shown in [13] and a 4 approximation for the weighted case is shown in [4]. To understand the difficulty of the subset selection problem, a relaxation of MEDP was proposed in [4]: to find a maximum fractionally routable subset of the demands. This *all-or-nothing* multicommodity flow problem (ANF) was studied in [5] where it was shown that the natural LP has a poly-logarithmic integrality gap. We mention that the inapproximability bound of [1] also applies to ANF, and hence the upper and lower bounds are separated by only a poly-logarithmic factor.

The focus of the present paper is a second relaxation of MEDP, that of allowing *congestion* on the edges of G. Here, we seek a sizable fraction of OPT by allowing the routable set to use each edge up to some constant number of times (alternatively, each edge is endowed with some constant capacity). Even with congestion c, it is shown in [2] that unless $NP \subseteq ZPTIME(n^{\text{polylog}(n)})$ no $O(\log^{\frac{1-\epsilon}{c+1}} n)$ approximation is possible. Similar lower bounds are also shown for the integrality gap of the flow relaxation [2]. However no super-constant integrality gap is known for planar graphs if congestion 2 is allowed. In contrast the best upper bound known was $O(\sqrt{n})$ until recent work [6, 7] which obtained a bound of $O(\log n)$ with congestion 2. Our main result, stated below, considerably strengthens this result.

THEOREM 1.1. *For the MEDP problem in a planar graph, there is a polynomial time algorithm to route $\Omega(\text{OPT})$ pairs with congestion 4, where OPT is the value of the multicommodity flow relaxation.*

We have given only the briefest overview to set the context and motivation for proving the above result. Some other related work in the area is discussed in the next subsection.

1.3 Outline

We outline the main ingredients of the proof Theorem 1.1, and how it differs from the framework used in [5, 6, 7] for solving the two throughput maximization problems ANF and MEDP. In the following we assume, without loss of generality, that the edge set F of the demand graph $H = (V, F)$ induces a matching M. We also let X denote the endpoints of M, the *terminals* of the instance.

Decompositions into Well-Linked Sets: Our previous approach to producing a large routable set was based on finding "well-connected" sets of terminals. A set X of terminals is *well-linked* if for each $S \subseteq V$ with $|S \cap X| \leq |(V \setminus S) \cap X|$, we have $|\delta(S)| \geq |S \cap X|$. The relevance of well-linked sets for edge-disjoint path problems in planar graphs is that a well-linked set implies the existence of a grid minor of size $\Omega(|X|)$ [28]. Hence, if edge congestion 2 is allowed, one may satisfy a constant fraction of any matching on the terminals, by routing through the grid minor [6].

We consider one approach to find a well-linked subgraph, starting from a fractional flow. We nominally treat terminals as having "weight" one, but in general this weight represents the total flow they originate in the fractional solution. We find an induced subgraph $G[S]$ that contains say p terminals from the pairs while $|\delta_G(S)|$ is small compared to p, say ϵp. We say S has a short boundary with respect to the terminals inside. Note that if the terminal set is not well-linked in the original graph, then there is a sparse cut S with respect to the terminals, and this identifies a short-boundary set. If we choose S carefully and the terminals in $G[S]$ turn out to be (approximately) well-linked, then the total flow lost by removing edges of $\delta(S)$, can be *charged* to $G[S]$ and we can recurse on $G[V \setminus S]$. This scheme can be applied to obtain a well-linked decomposition with only a constant factor loss if G happens to be a capacitated tree: choose S to be a *minimal* set with a short boundary. (This observation combined with Räcke's [25] hierarchical decomposition of graphs can be used to obtain a well-linked decomposition with a poly-logarithmic approximation ratio. This is implicitly shown in [5].) In general graphs or even planar graphs choosing

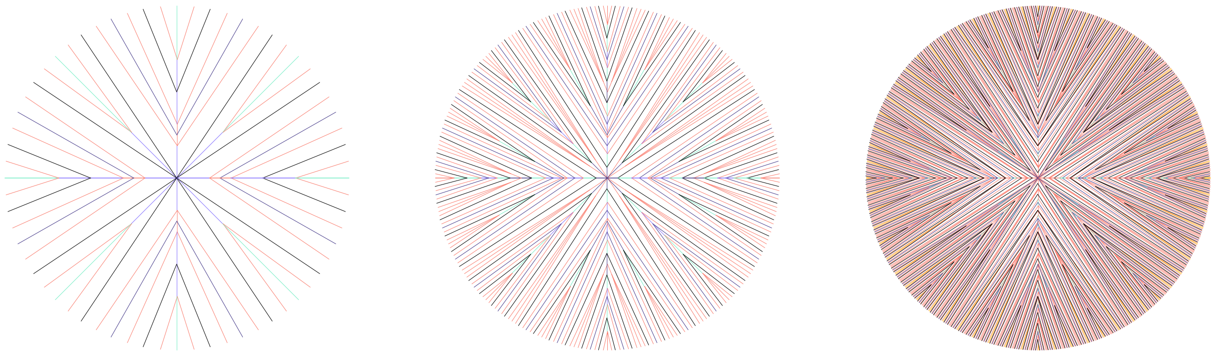

Figure 1: The duals D_6, D_8, D_{10} of the 6,8, and 10-dimensional recursively divisible planar graphs.

a minimal short boundary set S does not guarantee well-linkedness of the terminals in $G[S]$. In [7] a simple recursive scheme was used. For some ϵ that is sufficiently small (inverse poly-logarithmic), we find a set S that is ϵ-short and recurse on $G[S]$ and $G[V \setminus S]$.

It is natural to ask whether the above recursive procedure yields a well-linked decomposition that retains a constant factor of the flow if ϵ is some fixed constant, in particular for planar graphs. If this is the case, then we would obtain a constant factor approximation for MEDP in planar graphs with congestion 2. Unfortunately there are instances where using the recursive procedure (for any constant ϵ) will not retain even a logarithmic fraction of the initial flow. Such instances have a property we call *recursive divisibility* with respect to a set of terminals X. More concretely, there is a binary tree whose leaves are the nodes of V and for each internal node u, with subtrees T_l, T_r we have that the number of edges of G with one end in T_l and the other in T_r is at most $\epsilon \min\{|T_l \cap X|, |T_r \cap X|\}$. In other words, this tree represents a *sparse cutting procedure* where each cut produced along the way is "ϵ-sparse" relative to the original terminals. In Section A we show general graphs which are recursively divisible for $\epsilon = O(1/\log n)$, and planar graphs that are recursively divisible for any constant $\epsilon > 0$. These graphs suggest that a well-linked decomposition might result in super-constant approximation in planar graphs – see Figure 1.3 for a picture of the duals of graphs in this class. The precise definitions of a well-linked decomposition and the distinction between cut and flow well-linkedness can be found in Section A of the appendix. The following theorem shows a lower bound in general graphs; the appendix contains additional details.

THEOREM 1.2. *There are instances of EDP on a graph with n nodes, with* OPT *denoting the value of the multicommodity flow relaxation such that any cut-well-linked (and hence any flow-well-linked) decomposition retains at most $O(\frac{\text{OPT}}{\log n})$ pairs from the matching.*

Outline of the New Algorithm: The description above shows that in order to produce a constant factor result for MEDP in planar graphs, we might need to diverge from the framework of finding well-linked subgraphs. As before, we continue to seek a decomposition into "well-connected" pieces. However, our notion of well-connected is weakened considerably. In a way, this is not surprising since well-linked sets admit good routings for *any* matching on the terminals inside the well-linked piece $G[S]$. In the new framework, the definition of well-connected is more tied to the specific set of demands for an instance.

The new algorithm looks for a minimal subset S such that $|\delta(S)|$ is small (at most some constant factor $\epsilon < 1$) relative to the total fractional flow in $G[S]$. We then show how to pick a subset of the demands with both ends in S that can be routed. This is achieved as follows. First, using minimality of S one can show that terminals in S can be fractionally routed to the outside face; call this the *to-the-face routings*. Second, this routing is used to set up a feasible auxiliary multicommodity flow instance for pairs of nodes on the outside face of $G[S]$. The flow between distinct pairs are assorted values in the range $[0, 1]$. These are called OS instances (Okamura-Seymour). For the OS instance, we then solve an all-or-nothing flow problem, that is, for each pair its flow becomes either 0 or 1. We show that such a conversion is possible with only a constant factor loss in the total flow. We then apply the Okamura-Seymour theorem for the resulting pairs, to obtain a half-integral flow, thus corresponding to routing with congestion 2. Finally, this routing is stitched together with the to-the-face routing to satisfy a large fraction of our original demands. With some care, this can be achieved with an additional congestion of 2, to give the overall congestion bound of 4.

Other Related Work: Disjoint paths and routing problems have an extensive literature. We do not attempt to do justice in this extended abstract, and highlight only the most relevant work. We suggest [12, 29, 16, 20] for extensive introductions to the algorithmic work and some recent papers [6, 7, 17] for other pointers. The seminal work of Robertson and Seymour on graph minors, and their polynomial time algorithm for the disjoint paths problem for a fixed number of pairs [27], has had enormous impact on subsequent research. For planar graphs the first non-trivial algorithms for MEDP were given by Kleinberg and Tardos [18, 19] who considered grids and grid-like graphs. It led them to conjecture that poly-logarithmic approximation algorithms exist for all even-degree planar graphs. Implicitly the conjecture implied the existence of a poly-logarithmic approximation for planar graphs with congestion 2. This was achieved using well-linked decompositions introduced in [6, 7]. Kleinberg [17] obtained an $O(\log^2 n)$ approximation for all even-degree planar graphs also using such decompositions. In a variety of multicommodity flow problems Eulerian instances admit integral solutions where one might only suspect half-integral solutions – see [12, 29] for further details.

2. PRELIMINARIES

Simplifying the Input Instance: The input consists of an undirected planar graph $G = (V, E)$ with integer edge capacities and a set of node pairs $s_1t_1, s_2t_2, \ldots, s_kt_k$. We assume without loss of generality that each node in G has degree at most 4 and all edges have unit capacity. We refer the reader to [12, 6] for more details on how a given instance can be transformed to the bounded degree case while preserving planarity. The transformation increases the size of the graph to $\sum_e c(e)$. Note that this need not be strongly polynomial in the input size. However, as we indicate later, when using the LP relaxation, it is sufficient to assume $\sum_e c(e)$ is polynomial in n. We also assume that the node pairs are induced by a matching M on the terminal set X.

Multicommodity Flow LP Formulation: For the given instance with $\mathcal{T} = \{s_1t_1, s_2t_2, \ldots, s_kt_k\}$, we let $\mathcal{P}_i$ denote the paths joining s_i and t_i in G and let $\mathcal{P} = \cup_i \mathcal{P}_i$. The following multicommodity flow relaxation is used to obtain an upper bound on the number of pairs from $\mathcal{T}$ that can be routed in G.

$$\begin{aligned}
&\max \sum_{i=1}^{k} x_i \quad \text{s.t} \\
&x_i - \sum_{P \in \mathcal{P}_i} f(P) \;=\; 0 \qquad 1 \le i \le k \\
&\sum_{P: e \in P} f(P) \;\le\; c(e) \quad e \in E \\
&x_i, f(P) \;\in\; [0,1] \quad 1 \le i \le k, P \in \mathcal{P}.
\end{aligned}$$

For each path $P \in \mathcal{P}$ we have a variable $f(P)$ which is the amount of flow sent on P. We let x_i denote the total flow sent on paths for pair i. We let $\bar{f}$ denote the flow vector with a component for each path P, and we denote by $|\bar{f}|$ the value $\sum_i x_i$. We let OPT denote the value of an optimum solution to the relaxation. Call a path P *fractionally routed* if $f(P) \in (0,1)$, otherwise $f(P) \in \{0,1\}$ and P is *integrally routed.* If the total flow routed on integrally routed paths is more than OPT/2, then we already obtain a 2-approximation. Thus the interesting and difficult case is when the fractionally routed paths contribute almost all the value of OPT. From standard polyhedral theory the number of fractionally routed paths in a basic solution to the LP above is at most m. Therefore we can assume that $c(e) \le m$ for all edges. By making parallel copies of edges, in the following, we assume that G has only unit capacity edges. We can do the transformation to the bounded degree case that we mentioned above after solving the LP and this ensures that the resulting graph has size polynomial in n.

We work with a given fractional solution to the LP, not necessarily an optimum solution. We implicitly work with a flow-decomposition for the solution provided by the LP. Thus, when we remove edges in the graph, flow is "lost" on flow paths that use the removed edges. From a given solution, we define a weight function $b : X \to \mathcal{R}^+$ as follows: for a terminal $v \in X$, we let $b_v = x_i$ where $v \in \{s_i, t_i\}$. As the algorithm progresses, the b_v value might decrease as edges are removed and flow is lost. In our algorithm we set up some single source flow instances to route flow from terminals to other nodes. In these instances we have an upper bound of b_v on the flow we allow from a terminal v. In such a context we refer to b_v as its (meaning v's) "flow".

Planar Embeddings and Contours: We assume that we have some fixed embedding of G on the sphere. We also fix some point ι of the sphere. A *G-curve* is a simple curve in the plane, whose image only intersects the embedding of G at nodes. If the image of its "beginning" and "endpoints" are the same, then we call such a *closed* simple curve a *G-contour*, or just a contour. The *length* of a G-curve C, denoted by $len(C)$, is the number of nodes of G whose embedding is contained in the image of the curve. (In the following, we abuse terminology and do not differentiate between nodes, edges, and their images in the embedding.) For any contour C, removing its image from the sphere produces two open regions (disks). We let $ins(C)$ (respectively $ext(C)$) denote the region not containing ι (respectively containing ι). Without loss of generality, there is a G-contour C whose length is 0 and whose interior contains the embedding of G. For a contour C we use the notation G_C to denote the subgraph of G induced by the nodes in the closed interior of C. See Fig 2.

3. THE APPROXIMATION ALGORITHM

Given an instance of EDP on a planar graph G, we assume that we have performed the simplifications from the previous section. In particular, we assume we are given some optimal multicommodity flow $\bar{f}$. We then perform the simplifications

The heart of the approach is to find a contour C and its associated subgraph G_C with two properties. The first property is that the total flow from $\bar{f}$ contained inside G_C is at least a constant fraction of flow from $\bar{f}$ that is either transiting through G_C or crossing G_C. The second property is that the terminals in G_C can *simultaneously* send a constant fraction of their flow in $\bar{f}$ to the boundary C. We ensure the first property in a simple way by choosing C such that $len(C)$ is a sufficiently small factor γ smaller than the total flow incident to terminals in G_C. Since the degree of a node on C is at most 4, if γ is chosen small enough, the total flow inside G_C (that is, flow on paths that use only edges in G_C) is at least a constant factor larger than the flow crossing or transiting through G_C.

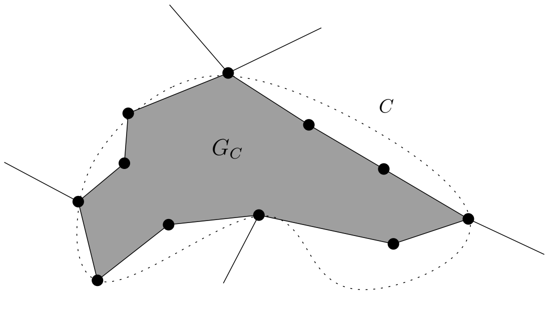

Figure 2: Contour

Suppose we find such a contour C. We restrict attention to the graph G_C and the flow from $\bar{f}$ that is induced in G_C. Using the second property, we set up a fractional instance of a disjoint path problem where all the terminals are on the outer face of G_C. We call such an instance an *Okamura-Seymour instance* (OS instance for brevity). For a pair st in G_C we set up a corresponding pair uv where $u, v \in C$. We show that for OS instances, the integrality gap of the

LP is a constant in the cardinality case. This is related to the pairing lemma of Frank [11] and strongly generalizes the result on trees [13]. The details are in Section 3.3. It is not sufficient to simply prove this. We also need to ensure that if a pair uv on C corresponding to a pair $s_i t_i$ is routed, then s_i and t_i can integrally route to u and v. To ensure this we set up the OS instance carefully, the details of which can be found in Section 3.2.

Thus we are able to integrally route (with constant congestion) a constant factor of the flow inside G_C. We then remove G_C from G and recurse on the remaining graph (and the induced flow). The first property of the contour C ensures that the total flow "lost" in removing G_C from G can be charged to the flow that is integrally routed in G_C. This process thus results in a constant factor approximation. Let $C_1, C_2, \ldots, C_\ell$ be the contours found in the process. The sub-graphs $G_{C_1}, G_{C_2}, \ldots, G_{C_\ell}$ are node-disjoint - in fact they are only edge-disjoint in the original graph before we do the transformation to the bounded degree case.

3.1 Finding and Routing to a Contour

In this section, we take as our starting point a fractional multicommodity flow vector $\bar{f}$ for the given instance such that for each commodity i, there is a total flow of x_i between s_i, t_i. For a terminal v let f_v denote the flow for the pair in which v participates. For any contour C, we denote by $T(C)$, the terminals that lie in G_C. We call a contour C *short* if its length (number of nodes) is at most $\lfloor \sum_{v \in T(C)} f_v/10 \rfloor$. Since maximum degree of any node is at most 4, it follows that for any short contour C, there is at least

$$\begin{aligned} \sum_{v \in T(C)} f_v/2 - 4len(C) &\geq \sum_{v \in T(C)} f_v/2 - 4 \sum_{v \in T(C)} f_v/10 \\ &\geq \sum_{v \in T(C)} f_v/10 \end{aligned}$$

flow entirely contained in G_C.

We call a short contour *good* if the terminals in $T(C)$ can each send $1/10$ of their flow to the contour. We make use of the following lemma to find a short good contour C.

LEMMA 3.1. *Let G be a planar graph embedded in the plane and let $w : V \to [0,1]$ with $w(V) > 1$. Given $\alpha \in (1, w(V))$, there is a contour C computable in polynomial time with the following properties:*

- *C is α-tight, that is $len(C) = \lfloor w(V(G_C))/\alpha \rfloor$.*
- *C is α-good, that is, in the graph G_C, each node u can simultaneously send $w(u)/\alpha$ flow to C such that a node in C receives at most 1 unit of flow.*

PROOF SKETCH. We say that a contour C is α-short if $len(C) \leq \lfloor w(V(G_C))/\alpha \rfloor$. Given a contour C that is α-short we can use simple greedy extension steps to find another contour C' that is contained in C and such that C' is α-tight.

Let C be an α-tight contour that minimizes the number of nodes in G_C. We claim that C is the desired contour. Suppose not. We consider a single-source flow problem $\mathcal{I}_C$ defined as follows. We create G' by adding to G_C, a new source node s and make it adjacent to each terminal v in the closed interior of C with edges of capacity $w(v)/\alpha$. We also add a node t and make it adjacent to each node v on C with a unit capacity edge. C is good iff the maximum s-t flow in G' equals $\sum_{u \in G_C} w(u)/\alpha$. If C is not good then the minimum s-t cut identifies a set of edges A whose removal results in a collection of disconnected components in G_C. One of these components, with node set S say, satisfies the property that the number of edges from A in $\delta_{G'}(S)$ is no more than $w(S)/\alpha$. This implies that there is a contour C' whose interior contains precisely the nodes of S and intersects S in at most $w(S)/\alpha$ nodes. Thus C' is also α-short. This contradicts the minimality of C since S must necessarily contain strictly fewer nodes from G_C.

The above proof can be made algorithmic in a straight forward fashion; either the current contour C is a short good contour in which case we stop, or when we solve I_C we find another short contour C' strictly contained inside C and we iterate. $\square$

We can use the above lemma with $w(v) = f_v$ if $v \in X$ and $w(v) = 0$ otherwise and set $\alpha = 10$. Let C be contour obtained; C is both a short and a good contour. The contour C contains two types of terminals. A terminal whose mate is outside is termed *separated* and we eliminate these. Let Q be remaining terminals; each terminal in Q has its mate also in Q (both are in G_C). Let R be the set of terminal pairs in G_C that are not separated. For a terminal $v \in Q$ let $b_v \leq f_v$ be the flow left in G_C. For a pair $st \in R$, let $b_{st} = b_s = b_t$ denote the flow between s and t that remains inside G_C. Let $p = \sum_{st \in R} b_{st} = \sum_{v \in Q} b_v/2$ be the total flow in G_C. As we have seen, $\sum_{v \in R} b_v/2 \geq \sum_{v \in T(C)} f_v/10$. Next we show that $\Omega(p)$ pairs from R can be routed in G_C.

3.2 Creating and Using the OS Instance

We assume that G_C is 2-node connected. If this is not the case, we can work with the block-tree of G_C and apply some ideas from routing on trees [13, 4] to reduce to the two-node connected case. We omit details in this version.

If G_C is two-node connected, then the outer face G_C is a cycle and C consists of a subset of nodes on the outer face; see Fig 2. We now create a fractionally feasible OS instance on G_C, that is, an instance of EDP where the terminals of the pairs are on the outer face C and we have a feasible fractional (partial) routing for demands between the pairs. Specifically, we create a set S of pairs of nodes chosen from C such that for each pair $uv \in S$, we have a flow $d(uv) \in [0,1]$, and these flows obey the edge capacities in G_C. Moreover, we ensure that $\sum_{uv \in S} d(uv) = \Omega(p)$. The pairs in S will serve as *surrogate pairs* for those in R, and we set up a mapping $\pi : R \to S$ with several properties that we describe later. In Section 3.3, we show that given a feasible multicommodity flow for an OS instance, there is a half-integral flow with only a constant factor loss in value. Thus, we find a subset $S' \subset S$ such that $|S'| = \Omega(p)$ and pairs in S' are routed integrally with congestion 2.

The idea is then to find a routable subset $R' \subset R$ by using S' and the mapping π. We do this as follows. For each $uv \in S'$ we add any pair $st \in \pi^{-1}(uv)$ to R'. To argue that R' is routable in G_C, we exhibit a half-disjoint path collection $\mathcal{P}$ such that the end points of each pair $st \in R'$ are joined to the end points of $\pi(st) \in S'$. This requires that we set up the OS instance S and the mapping π in a careful fashion. The above description shows that we obtain a set $R' \subset R$ such that $|R'| = \Omega(p)$ and R' is routable with congestion 4. We now describe details of the OS instance created.

For a set of nodes $A \subseteq V(G_C)$ let $b(A) = \sum_{v \in A \cap Q} b_v$ denote the total b value of terminals in A. We assume that $p \geq 10$ for otherwise we can route one pair and obtain a constant factor approximation. Our first goal is to partition the graph G_C into node-disjoint connected induced graphs $G[V_1], G[V_2], \ldots, G[V_h]$ such that for $1 \leq i \leq h$, $10 \leq b(V_i) \leq 10\Delta$. We do this by partitioning via a simple greedy procedure. This ensures that $h \geq \lfloor 2p/(10\Delta) \rfloor$. Here Δ is the maximum degree of the graph. In our instances $\Delta \leq 4$, however it is easier to understand the description with Δ in place of the constant 4.

Find a spanning tree T of G_C and root it at some node r. If $b(V) \leq 10\Delta$ stop and output $V_1 = V(T)$. Otherwise, find the deepest node v in T such that the subtree of T rooted at v, T_v, has total b-weight at least 10. We set $V_1 = V(T_v)$ and remove it from T. Note that $b(V(T_v)) < 10(\Delta-1)+1 \leq 10\Delta$ if $v \neq r$. We continue the process by finding another deepest node in the remaining tree and remove it and so on. At the end if we have a tree with total weight less than 10 we simply add it to the tree found in the previous step. This clustering step is similar to that used in [5].

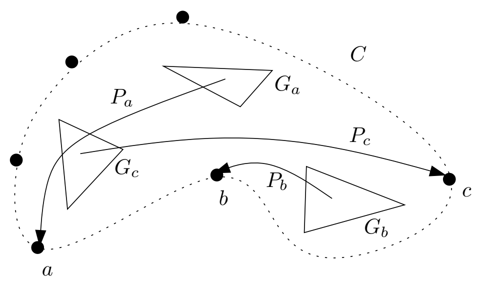

Figure 3: Setting up the OS instance.

Since every terminal v can simultaneously send $b(v)/10$ flow to the contour C, the node sets $V_1, V_2, \ldots, V_h$ can simultaneously send one unit of flow each to the contour C. It follows from the integrality of single source flows that there are edge-disjoint paths $P_1, P_2, \ldots, P_h$ where P_i has one end point on C and the other in V_i and further a node in C is the end point of at most one of these paths. Let $C' \subset C$ be the collection of end points of the paths in $\mathcal{P} = \{P_1, P_2, \ldots, P_h\}$. By construction, we have a bijection $g : C' \rightarrow \{1, 2, \ldots, h\}$. For a node $a \in C'$ we refer to $G[V_{g(a)}]$, for simplicity of notation, as simply G_a and the path $P_{g(a)}$ by P_a. See Figure 3.2 for an illustration.

Now we are ready to define S and set up the mapping π from R to S. Let u, v be distinct nodes on C'. We add the pair uv to S and set

$$d(uv) = \frac{1}{3 \cdot 10 \cdot \Delta} \sum_{st \in R, s \in G_u, t \in G_v} b_{st}.$$

For a pair $st \in R$ with $s \in G_u, t \in G_v$ we let $\pi(st) = uv$. First we note that for each $u \in C'$, $D(u) = \sum_{v \in C'} d(uv) \leq 1/3$. This follows from the fact that $b(V_i) \leq 10 \cdot \Delta$ for $1 \leq i \leq h$.

LEMMA 3.2. *The multicommodity flow instance on S with demands d is feasible in G_C.*

PROOF. Consider three identical copies of G_C, H_1, H_2, H_3. The capacity of an edge in H_i is a third of the capacity of its corresponding edge in G. We argue for the feasibility of d by considering some node $u \in C'$ and how it distributes its flow. The node u sends $D(u)$ flow along path P_u to the copy of G_u in H_1. We note that the path collection $\mathcal{P}$ is edge-disjoint in G_C and hence these flows are feasible in H_1 simultaneously for all nodes in C'. Let u' be the other end point of P_u. The node u' distributes $D(u)$ flow to the terminals in the copy of G_u in H_2. A terminal $s \in G_u$ is sent $\frac{1}{3 \cdot 10 \cdot \Delta} b_s$ flow. This can be done since G_u is connected and $\sum_{s \in G_u} b_s \leq 10\Delta$. Since the G_u's are disjoint, this distribution can be done simultaneously in H_2 for all such u' nodes. Now, the pairs in R use H_3 to route their multicommodity flow. This is feasible since the pairs had a feasible multicommodity flow in G_C and the flow is now scaled down by a factor of 3 at least. Composing these flows shows that d is feasible. □

We assume that for each $uv \in S$, $d(uv) > 0$. Otherwise we can remove uv from S.

LEMMA 3.3. *Let $S' \subseteq S$ such that each node $u \in C'$ is incident to at most one pair in S'. Then there is a $R' \subset R$ and a path collection $\mathcal{Q}$ with the following properties: (i) $|R'| = |S'|$ and $\pi(R') = S'$, (ii) for each pair $st \in R'$, there are two paths $\mathcal{Q}_s$ and $\mathcal{Q}_t$ in $\mathcal{Q}$ that originate at s and t respectively and end at $\pi(s)$ and $\pi(t)$, and (iii) an edge e in G_C is in at most two paths in $\mathcal{Q}$.*

Note that in the above lemma Q_s might end at $\pi(t)$ and Q_t might end at $\pi(s)$.

PROOF. We create R' from S' as follows. For each $uv \in S'$ we pick an arbitrary but a single pair $st \in \pi^{-1}(uv)$ and add st to R'. Such a pair must exist if $d(uv) > 0$. Therefore $|R'| = |S'|$ and $\pi(R') = S'$ as required. By construction of S we have that $s \in G_u$ and $t \in G_v$. We now create the paths $\mathcal{Q}_s$ and $\mathcal{Q}_t$. Let u' and v' be the end points of P_u and P_v in G_u and G_v respectively. Then $\mathcal{Q}_s$ is the path obtained by composing a path from s to u' in G_u and the path P_u. Similarly, for $\mathcal{Q}_t$. We observe that the path collection $\mathcal{P}$ is edge-disjoint and the graphs $G_1, \ldots, G_h$ are node-disjoint and hence an edge can participate at most twice in the path collection $\mathcal{Q}$ created as above. □

Although the OS instance on S is well defined, it could be the case that $\sum_{uv \in S} d(uv)$ is significantly smaller than p, the total flow that we started with in G_C. We argue that this is the easy case.

LEMMA 3.4. *If $\sum_{uv \in S} d(uv) \leq \frac{1}{6 \cdot 10 \cdot \Delta} p$ then $\Omega(p)$ pairs from R can be routed by edge-disjoint paths in G_C.*

PROOF. We call a pair $st \in R$ to be *distant* if $s \in V_i$ and $t \in V_j$ and $i \neq j$. Let A be the set of distant pairs. Call a set V_i *good* if there is a pair $st \in R$ with $s, t \in V_i$. By construction we have that

$$\sum_{uv \in S} d(uv) = \frac{1}{3 \cdot 10 \cdot \Delta} \sum_{st \in A} b_{st}.$$

Therefore, if $\sum_{uv \in S} d(uv) \leq \frac{1}{6 \cdot 10 \cdot \Delta} p$, a large fraction of the pairs in R are not distant. It follows that the total number of good sets in this case is $\Omega(p/(10 \cdot \Delta))$. We can route one pair in each good set and the sets are node-disjoint and hence the routed pairs use edge-disjoint paths. □

3.3 Choosing the Demands in an OS instance

In this section we consider OS instances: instances in which the terminals are on the outer face of a planar graph. The well-known Okamura-Seymour theorem states that the cut condition is sufficient for a half-integral flow. Here we are interested in the maximization version where not all pairs maybe routable. Even on trees (which is very restricted OS instance) it is known that the maximization problem is APX-hard [13]. We solve the LP for the problem which gives an upper bound and then show that we can recover a constant fraction of the LP solution value. We make some preliminary observations that lead us to set up the problem as an abstract ring routing problem. As we remarked earlier, we assume that the given graph is two-node connected and hence the outer face can be assumed to be a connected ring. For an OS instance we can focus on a restricted set of cuts given by pairs of edges on the outer face. Let e and e' be two edges on the outer face. Consider the cheapest cut in the graph that contains e and e' and let its value be $\mu(e,e')$: this can be obtained by a shortest path computation in the dual of the embedded graph. For an OS instance to be feasible, it suffices to check the feasibility of the given instance with respect to only the above types of cuts (see [24]). Using this, we set up our problem as a ring-routing problem below.

In the following, let $G = (V = \{0, 1, 2 \ldots, n-1\}, E)$ be an undirected ring with edges $e_i = i(i+1)$ for $0 \leq i < n$. In addition, we assume that for each pair of edges e_i, e_j, there is an associated integer capacity $\mu(e_i, e_j)$, or simply $\mu(ij)$. We also consider a *demand multiset* F of edges with endpoints in V. Our sets may also be *weighted* in that for each $f \in F$ there is a nonnegative demand value $d_f \in [0,1]$. The weight of such a set is then $d(F) = \sum_{f \in F} d_f$. An edge f *crosses* a pair e_i, e_j if the endpoints of f lie in distinct components of $G - \{e_i, e_j\}$. The *load* on a pair (cut) e_i, e_j is just the sum of the demands that cross the pair, i.e., $\sum_{f \in F(ij)} d_f$ where $F(ij)$ are those demands that cross e_i, e_j. A weighted set of demands is *feasible* (*almost-feasible*) if for each e_i, e_j we have that its load is at most $\mu(e_i, e_j)$ (resp. $\mu(e_i, e_j) + 2$).

Our main result in this section is that if we start with a feasible fractional set of demands, then we may find a large integral set of demands that obey the "abstract" 2-cut conditions imposed by μ. A similar problem (the Pairing Lemma) was solved by Frank [11]. In a certain setting, Frank gave necessary and sufficient conditions for the existence of a perfect matching on V that obeys the μ-cut condition. A key difference is that Frank works with the complete graph, that is, his matching may use any edge ij, while we may select only from F.

THEOREM 3.5. *For any ring G, μ and feasible weighted set of demands F, d, there is an almost-feasible subset F' such that each demand in F' is a unit demand, and $|F'| \geq \alpha \sum_{f \in F} d_f$ for some fixed constant $\alpha > 0$.*

The rest of the section is devoted to the proof of the above theorem.

For a node $v \in V$ we let $w(v) = \sum_{f: v \in f} d_f$ denote the total weight of demand edges incident to v. For a set of nodes S we let $w(S) = \sum_{v \in S} w(v)$. We ensure that in the final solution the number of demands incident to any node of V is no more than $\lceil w(v) \rceil$. We assume without loss of generality that $w(v) \leq 1$. Otherwise we can split a node into multiple nodes connected by an infinite capacity edge.

In the following we use γ for a fixed integer that is at least 6. A *block* in our instance is a contiguous interval of nodes S on the ring such that $\gamma \leq w(S) < 2\gamma + 1$. At any given time we are working with a partition of the nodes into blocks $B_1, B_2, \ldots, B_h$ and so $h = \Omega(\sum_f d_f)$. Also, for a node u, we denote by $B(u)$ the block containing u. We start with a feasible blocking obtained by a simple greedy procedure that goes clockwise around the ring picking minimal blocks to satisfy the requirement that $\gamma \leq w(B_i) < 2\gamma + 1$. The blocks we create are intervals on the ring and there is a natural notion of adjacency between them.

We proceed to generate our feasible set of demands by repeating three operations: *augmenting*, *locking* and *reblocking*. An augmentation of a demand edge f involves pushing up its value d_f to 1; at the same time we sacrifice an $O(1)$ amount of other demands to relieve the load on cuts that f crosses. The demand $f = uv$ is then selected for F' and $B(u), B(v)$ are locked, and all demands with an endpoint in $B(u)$ or $B(v)$ are deleted. Once a block is locked no future blocks can contain any node from it.

Reblocking: We first describe the reblocking procedure which is used after any operation which deletes edges from our instance, and thus may produce blocks that are too light, i.e. of weight less than γ. Suppose that B is such a block and let B_l, B_r be its neighbouring blocks. If either B_l, B_r is not yet locked, say B_l, then we merge B and B_l. If the weight of this merged block is more than $2\gamma + 1$, then we use splitting to create two blocks. Note that this does not affect the weight of any of the other blocks. If both B_l and B_r are *locked*, then we delete the edges incident to B and charge them to B_l, B_r. Note that each locked block could get charged this way by at most one block on either side, and since every locked block contributes a routed demand to F', we lose at most a constant fraction of demand in this way. We call the block B *dead* at this point; it will not participate any more. A block that is neither dead nor locked is called a *live* block. In doing this, however, other light blocks may be created since we delete the demands incident to B. We repeat this process until all remaining live blocks are again of weight at least γ and at most $2\gamma + 1$.

Augmenting: We next describe the augmenting procedures. Each of these procedures consists of choosing a demand edge f and increasing d_f to 1 and adding it to F'. The blocks that contain the end points of f (could be a single block or a pair of blocks) are then locked and all demand edges incident to those blocks are deleted. After this we do reblocking as necessary and repeat the augmentation process until we have no more live blocks left. We now describe how to pick a demand f.

First, if there is some block B with a demand edge f with both ends in B, then we pick it. If not, we check if there are two distinct blocks B and B' such that the total weight of demands between them is at least 1. If so, we pick an arbitrary demand f between them. Call the two augmentations above *simple*. Note that if no simple augmentations exist then each block B has demand edges to at least γ other distinct blocks.

We next describe a more involved augmenting procedure. The intuition behind the procedure is as follows. Consider any pair of demand edges uv, vw where say $u < v < w$. If u has a neighbour z in the interval $[v, w]$ then we can bump up uv by some ϵ and reduce uz and vw by ϵ. We look for several simultaneous such augmentations for uv that can be used to increase its demand to 1 in one shot and select the demand edge uv for routing. Identifying such a demand edge is the key.

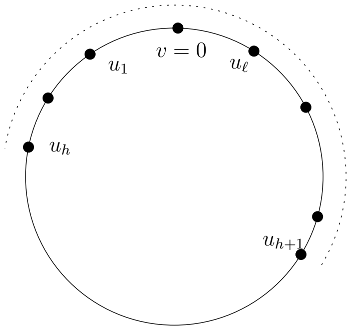

Figure 4: A cover for node v.

Let $B_0, B_1, \ldots, B_{n'-1}$ be the current set of live blocks (not locked or dead). We create an undirected ring $G' = (V' = \{0, 1, \ldots, n'-1\}, E')$ with the nodes corresponding to the blocks. Let $H = (V', X)$ be a demand graph induced on the blocks where $uv \in X$ if there is some demand edge f incident to nodes in B_u and B_v. We also associate weights with edges in X. We abuse notation and use d for this. The weight of uv, $d(uv)$ is the total sum of weights of demand edges from F with one end point in B_u and the other in B_v. If no simple augmentation is feasible then $d(uv) < 1$ and the degree of a node in H is at least $\gamma + 1$. We now work on the ring G'. We say a contiguous segment A of the ring "covers" node a if it contains a and $\sum_{u \in A} d(ua) \geq 4$. Choose a node v with a cover S of minimum length (in terms of nodes). Without loss of generality, assume $v = 0$ and v's neighbours on the segment S are $u_1 < u_2, \ldots < u_\ell$ numbered in anti-clockwise direction. The segment itself is the union of a "left" segment $L = [0, u_h]$ and a right one $R = [u_{h+1}, 0]$, i.e., S is the segment $[u_h, u_h - 1, \ldots, v = 0, n'-1, n'-2, \ldots, u_{h+1}]$. See Figure 4. Without loss of generality, $\sum_{a \in L} d(av) \geq 2$. Since we had no simple augmentations available, we may deduce that $\sum_{k=2}^{h} d(vu_k) \geq 1$ and $h \geq 3$.

Claim 3.6. *For $\gamma \geq 6$ we have $\sum_{y=u_h}^{n'-1} d(u_1 y) \geq 2$ and $\sum_{y=u_h+1}^{n'-1} d(u_1 y) \geq 1$.*

Proof. If $\sum_{y=u_h}^{n'-1} d(u_1 y) < 2$, segment $[0, u_h - 1]$ would be a cover for u_1 and contradict the choice of S as a min length cover. Since we had no simple augmentation, $d(u_1 u_h) < 1$, and hence $\sum_{y=u_h+1}^{n'-1} d(u_1 y) \geq 1$. $\square$

Let f be an arbitrary demand edge in G between blocks B_v and B_{u_1}. We choose f for augmentation. This finishes the description of the augmenting procedures and the algorithm. Let F' be the set of demands chosen during the course of the algorithm. We prove two properties of F'.

Lemma 3.7. *There is a constant α (that depends on γ) such that $|F'| \geq \alpha \sum_{f \in F} d_f$.*

Proof. We have argued that an augmentation can be done as long as there are live blocks, therefore at the end of the algorithm there are no live blocks left. We show the lemma by a charging argument. Each augmentation results in a demand f being added to F'. Let B_1 and B_2 be the blocks that contain the end points of f (in case f has end points in the same block then $B_1 = B_2$). These blocks are locked and all demand edges incident to those blocks are deleted. We charge the weight of the deleted demands to f. Recall that the weight of a block is at most $2\gamma + 1$. We associate f with the locked blocks B_1 and B_2. During reblocking operations we might charge a dead block B to a previously locked block such as B_1. However, as we argued, at most two such blocks can be charged to B_1 one from its "left" and one from its "right". Also, while charging only half their weight gets charged to B_1 since a dead block has two locked blocks on either side and can charge equally to them. We charge the weight of demands incident to these dead blocks to f. Thus a demand from F' gets a charge of at most $4\gamma + 2$. Thus we have that $|F'| \geq \frac{1}{4\gamma+2} \sum_{f \in F} d_f$. $\square$

Lemma 3.8. *The set F' is an almost-feasible set of demands for G, μ.*

Proof. Let e_i and e_j be two edges of the ring. The load put by F on the cut defined by e_i, e_j is $\sum_{f \in F(ij)} d_f$. Let $l(ij)$ denote this load. By the feasibility of F we have that $l(ij) \leq \mu(ij)$ for all i, j. We say that an edge e_i belongs to a locked block B if both end points of e_i lie in B. Let $z_i = 1$ if e_i lies in a locked block and 0 otherwise. If $z_i = 1$ we let f_i be the unique edge in F' associated with the locked block containing e_i. Let $A(ij)$ be the set of demands in $F'(ij)$ after removing f_i and f_j. We show that $|A(ij)| \leq \lfloor l(ij) \rfloor$. Note that $|F'(ij)| \leq |A(ij)| + 2$.

Consider an edge f in $A(ij)$. We observe that f can be added to F' either in second type of augmentation or the third type but not the first. Let B_1 and B_2 be the two blocks that f is incident to. Since e_i and e_j do not belong to either B_1 or B_2 (otherwise f would not be in $A(ij)$), all demands incident to B_1 and B_2 cross e_i and e_j. Consider the second type of augmentation. In this case the total weight of demands between B_1 and B_2 is at least 1. We add f and delete all demand edges incident to them - therefore we can charge f to the load of the deleted demands.

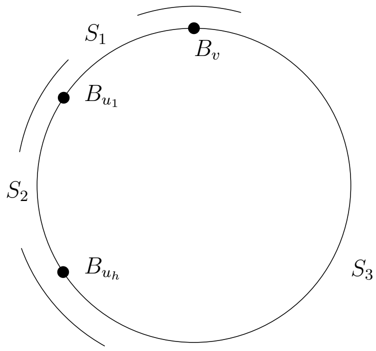

Figure 5: Analysis of augmentation.

Now we consider the third type of augmentation. In this case the weight of demands between B_1 and B_2 can be small and we need to use the careful choice of the demand f. We refer to the notation used when describing this type of augmentation. We have that f is an edge between block B_v

and B_{u_1}. We consider three segments, S_1, S_2, S_3, of the ring G obtained by removing the edges in the blocks B_v, B_{u_1} and B_{u_h}. S_1 is the segment between v and u_1, S_2 is the segment between u_1 and u_h and S_3 is the segment between u_h and v. See Figure 5. We assume without loss of generality the edge e_i is in the segment S_1. We consider two cases based on the location of e_j. If e_j is in S_2 or B_{u_h} we charge f to the load of the demand edges from B_{u_1} to the segment S_3. From Claim 3.6 this load is at least 1. If e_j is in S_3 we charge f to the load of edges from B_v to S_2 and B_{u_h}. This load is at least 2 as we noted earlier. □

Lemmas 3.7 and 3.8 finish the proof of Theorem 3.5.

Now we come back to OS instances. Recall that we assumed that the graph is two-node connected and hence the outer face is a cycle. From this it follows that $\mu(ij) \geq 2$ for any two edges e_i and e_j on the ring formed by the outer face. From Lemma 3.8 we see that the set of demands in the final solution F' satisfies the condition that $|F'(ij)| \leq 2 + \lfloor l(ij) \rfloor$. By initially scaling down the demand values, we can assume that $l(ij) < \mu(ij)/2$. Therefore it follows that $2 + \lfloor \mu(ij)/2 \rfloor \leq \mu(ij)$ and hence $|F'(ij)| \leq \mu(ij)$. We thus obtain the following theorem.

THEOREM 3.9. *For OS instances on two-node connected planar graphs, the integrality gap for the cardinality version of the ANF problem is $O(1)$.*

4. CONCLUDING REMARKS

We believe that an $O(1)$ approximation for MEDP in planar graphs can be obtained with congestion 2 instead of 4. It seems feasible to extend our approach to graphs with bounded genus and to graphs that exclude a fixed minor. We plan to explore this in future work.

Acknowledgments: Sanjeev Khanna is supported in part by an NSF Career Award CCR-0093117. Chandra Chekuri and F. Bruce Shepherd acknowledge support from an ONR basic research grant N00014-05-1-0256 to Lucent Bell Labs.

5. REFERENCES

[1] M. Andrews and L. Zhang. Hardness of the undirected edge-disjoint paths problem. *Proc. of ACM STOC*, 2005.

[2] M. Andrews, J. Chuzhoy, S. Khanna and L. Zhang. Hardness of the undirected edge-disjoint paths problem with congestion. *Proc. of IEEE FOCS*, 2005.

[3] Y. Aumann and Y. Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. on Comp.*, 27(1):291–301, 1998.

[4] C. Chekuri, M. Mydlarz, and F. B. Shepherd. Multicommodity Demand Flow in a Tree and Packing Integer Programs. *Proc. of ICALP*, 2003.

[5] C. Chekuri, S. Khanna, and F. B. Shepherd. The All-or-Nothing Multicommodity Flow Problem. *Proc. of ACM STOC*, 2004.

[6] C. Chekuri, S. Khanna, and F. B. Shepherd. Edge-Disjoint Paths in Planar Graphs. *Proc. of IEEE FOCS*, 2004.

[7] C. Chekuri, S. Khanna, and F. B. Shepherd. Multicommodity Flow, Well-linked Terminals, and Routing Problems. *Proc. of ACM STOC*, 2005.

[8] C. Chekuri, S. Khanna, and F. B. Shepherd. An $O(\sqrt{n})$ approximation and integrality gap for EDP and UFP in undirected graphs and DAGs. Submitted, September 2005.

[9] S. Even, A. Itai and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM J. on Computing*, Vol 5, 691-703, 1976.

[10] L. R. Ford, D. R. Fulkerson, 1962. Flows in Networks. Princeton University Press, Princeton, NJ.

[11] A. Frank. Edge-disjoint paths in planar graphs. *J. of Combinatorial Theory, Ser. B.*, No. 2, 164–178, 1985.

[12] A. Frank. Packing paths, cuts, and circuits - a survey. In B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, eds., *Paths, Flows and VLSI-Layout*, 49–100. Springer Verlag, Berlin, 1990.

[13] N. Garg, V. Vazirani, M. Yannakakis. Primal-Dual Approximation Algorithms for Integral Flow and Multicut in Trees. *Algorithmica*, 18(1):3–20, 1997.

[14] R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, R. E. Miller, J. W. Thatcher, Eds., New York: Plenum Press, 1972, 85–103.

[15] P. Klein, S. Plotkin and S. Rao. Planar graphs, multicommodity flow, and network decomposition. *Proc. of ACM STOC*, 1993.

[16] J. M. Kleinberg. Approximation algorithms for disjoint paths problems. PhD thesis, MIT, May 1996.

[17] J. M. Kleinberg. An Approximation Algorithm for the Disjoint Paths Problem in Even-Degree Planar Graphs. *Proc. of IEEE FOCS*, 2005.

[18] J. M. Kleinberg and É. Tardos. Approximations for the disjoint paths problem in high-diameter planar networks. *JCSS*, 57:61–73, 1998.

[19] J. M. Kleinberg and É. Tardos. Disjoint Paths in Densely Embedded Graphs. *Proc. of FOCS*, 1995.

[20] S. G. Kolliopoulos. Edge Disjoint Paths and Unsplittable Flow. *Handbook on Approximation Algorithms*, Chapman Hall/CRC Press, to appear.

[21] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *JACM*, 46(6):787–832, 1999. Prelim. version in *Proc. of IEEE FOCS*, 1988.

[22] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

[23] K. Menger. Zur Allgemeinen Kurventheorie Fundam. Math. **10**, 96–115, 1927.

[24] H. Okamura and P.D. Seymour. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory, Series B*, 31, 75–81, 1981.

[25] H. Räcke. Minimizing congestion in general networks. *Proc. of IEEE FOCS*, 2002.

[26] S. Rao. Small distortion and volume preserving embeddings for planar and Euclidean metrics. *Proc. of SoCG*, 300–306, 1999.

[27] N. Robertson and P. D. Seymour. Outline of a disjoint paths algorithm. In B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, Eds., *Paths, Flows and VLSI-Layout*. Springer-Verlag, Berlin, 1990.

[28] N. Robertson, P. D. Seymour and R. Thomas. Quickly Excluding a Planar Graph. *Journal of Combinatorial*

Theory (B), 62: 323-348, 1994.
[29] A. Schrijver. Combinatorial Optimization: Polyhedra and Efficiency. Springer-Verlag, 2003.

APPENDIX

A. WELL-LINKED DECOMPOSITIONS AND RECURSIVELY DIVISIBLE GRAPHS

We now discuss some lower bounds on obtaining approximations for EDP and ANF via the approach of well-linked decompositions. For us a well-linked decomposition is a partition of the graph into node disjoint induced subgraphs such that each induced subgraph retains a collection of well-linked terminals. We say a terminal is *retained* in the decomposition if it is part of the specified (flow)-well-linked set in one of the subgraphs. We are interested in the ratio of the total number (or more generally, total fractional weight) of terminals retained as a fraction of OPT, the value of an optimum solution to the flow relaxation. We distinguish between flow and cut well-linkedness. We mention that flow-well-linkedness is a stronger notion than cut-well-linkedness and are approximately related by $\beta(G)$ the product-multicommodity flow-cut gap in G. We refer the reader to [7] for details on these. For planar graphs $\beta(G) = O(1)$ [15] and hence flow and cut-well-linkedness are roughly equivalent. Let $\gamma_f(G)$ and $\gamma_c(G)$ denote the largest fraction of terminals that can be retained, as a fraction of OPT, in the worst-case in a flow-well-linked decomposition and cut-well-linked decomposition respectively. In [7] it is shown that $\gamma_f(G) = \Omega(1/(\beta(G)\log n))$ where $\beta(G)$ is the worst-case bound on the flow-cut gap for product multicommodity flows in G and that $\gamma_c(G) = \Omega(1/\log n)$. [1]

Note that if terminals are flow-well-linked, then we obtain a constant fraction approximation for ANF. Hence it follows that $\gamma_f(G)$ is upper bounded by the integrality gap of the flow relaxation for ANF. From the results in [2] we have that $\gamma_f(G) = O(1/\sqrt{\log n})$ for general graphs. For EDP in planar graphs we can achieve an $O(1/\gamma_c(G))$ approximation with congestion 2 [6]. Here we show that for general graphs $\gamma_c(G) = O(1/\log n)$ and hence $\gamma_f(G) = O(1/\log n)$. We also describe planar graph instances for which we believe $\gamma_c(G) = o(1)$.

Upper bound for $\gamma_f(G)$ and $\gamma_c(G)$ in general graphs

Let H_k denote the standard k-dimensional hypercube. We create a routing instance on the hypercube as follows. For each edge $e = uv$ in H_k we have a demand pair uv. Clearly these demands can be routed in H_k in edge-disjoint fashion. We can extend this to be in "standard form", where each terminal participates in exactly one demand pair. We modify the graph by attaching dummy terminals to vertices in H_k. In particular, we attach to each node u in H_k, k additional nodes $u_1, u_2, \ldots, u_k$ connected to u, their *parent*, in the form of a star. For an edge $e = uv$ in H_k, there is now a demand pair $u_i v_i$ if u and v differ in the i^{th} bit. Let M denote the matching to be routed, that is, the set of demand pairs formed above. Also, let $n = 2^k$ denote the total number of nodes in the hypercube H_k, and let H'_k denote the new instance with $N = kn$ nodes.

[1]The best bound that can be achieved in polynomial time is $\Omega(1/(\beta(G)\log n))$ where $\beta(G)$ is the best approximation ratio for finding sparse cuts.

Since the above instance is (integrally) feasible, there is a total flow on H'_k of size $k2^{k-1} = \Theta(N)$. It is clear that there is a flow-well-linked decomposition that retains $\Omega(N/\log N)$ flow: simply pick any matching in H_k, each edge being a well-linked set. The theorem below shows that this is best possible for even a cut-well-linked decomposition. We omit the proof in this version.

THEOREM A.1. *Any cut-well-linked decomposition of the instance above retains at most $O(N/\log N)$ edges from the matching M.*

Recursively Divisible Planar Graph Instances

The hypercube example shows that we may not generally expect (much) better than a logarithmic fraction of the flow from an LP solution, to be retained in a well-linked decomposition. One may hope for a rosier picture in the case of planar graphs. Ostensibly it may be possible to even retain a constant fraction of the demand.

Related to our comments in Section 1.3 we describe a class of planar graphs that are bad for recursive partitioning algorithms if the partition procedure uses a particular sequence of ϵ-sparse cuts for some fixed constant $\epsilon > 0$. These graphs may well have the stronger property that $\gamma_c(G) = o(1)$. Although we do not have concrete evidence for this, (the structure of the graphs is still a bit mysterious and fascinating to us), we think that these graphs are suggestive of a lower bound.

We now define a class of recursively divisible planar graph instances denoted by P_n. For any constant ϵ, these are graphs such that if we repeatedly split along ϵ-sparse cuts, then we end up with a collection of subgraphs that retain very little of the initial flow. It is easier to view how these graphs arise by looking at their planar dual, denoted by D_n. We suppose at first that $\epsilon = 1$. Each edge of D_n has an associated length $l(e)$. The graph D_1 can be thought of as a circle C of length 1 with 2 nodes u, v at antipodal points on the circle. There is also another edge of length 1/2 joining u, v. We now recursively define D_{n+1} from D_n as follows. At any point, D_n is outerplanar and the boundary of each face F can be written as $C_F \cup I_F$, where C_F is a segment of the original cycle C, and I_F is a curve inside C. Let i_F be the midpoint (with respect to the length function l) of the curve I_F (note that there may be no node at this midpoint). Let c_F denote the midpoint of C_F. In D_{n+1}, we add the nodes i_F, c_F, and join them by an edge of length $\frac{1}{2^n}$. In Figure 1.3, the graphs (without lengths) D_6, D_8, D_{10} are shown.

The recursively divisible graph P_n is obtained by drawing the dual of D_n and removing the node corresponding to the infinite face and its incident edges. For an edge $uv \in D_n$, it contributes $2^n l(uv)$ parallel edges in P_n. With each edge in P_n we associate a demand pair in the routing instance and hence the total flow routed is the same as the total number of edges in P_n. The recursive definition of D_n gives rise naturally to a sparse cutting procedure in P_n. At the i^{th} level, there are 2^i faces in D_n, each containing 2^{n-i} nodes of P_n. Each such face is then split in half and the two halves are joined by 2^{n-i} edges. We ensure ϵ-sparseness by adding $1/\epsilon$ terminals at each node in P_n. Thus for any fixed ϵ, a recursive procedure that partitions along these ϵ-sparse cuts, removes almost all the edges of P_n and hence the associated demands.