Improved Approximation for Node-Disjoint Paths in Planar Graphs*

Julia Chuzhoy†
Toyota Technological Institute at Chicago
6045 S. Kenwood Ave
Chicago, IL 60637, USA
cjulia@ttic.edu

David H. K. Kim‡
Computer Science Department
University of Chicago
1100 E. 58th Street
Chicago, IL 60637, USA
hongk@cs.uchicago.edu

Shi Li§
Department of Computer Science and Engineering
University at Buffalo
338 Davis Hall
Buffalo, NY 14260, USA
shil@buffalo.edu

ABSTRACT

We study the classical Node-Disjoint Paths (NDP) problem: given an n-vertex graph G and a collection $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of pairs of vertices of G called *demand pairs*, find a maximum-cardinality set of node-disjoint paths connecting the demand pairs. NDP is one of the most basic routing problems, that has been studied extensively. Despite this, there are still wide gaps in our understanding of its approximability: the best currently known upper bound of $O(\sqrt{n})$ on its approximation ratio is achieved via a simple greedy algorithm, while the best current negative result shows that the problem does not have a better than $\Omega(\log^{1/2-\delta} n)$-approximation for any constant δ, under standard complexity assumptions. Even for planar graphs no better approximation algorithms are known, and to the best of our knowledge, the best negative bound is APX-hardness. Perhaps the biggest obstacle to obtaining better approximation algorithms for NDP is that most currently known approximation algorithms for this type of problems rely on the standard multicommodity flow relaxation, whose integrality gap is $\Omega(\sqrt{n})$ for NDP, even in planar graphs. In this paper, we break the barrier of $O(\sqrt{n})$ on the approximability of NDP in planar graphs and obtain an $\tilde{O}(n^{9/19})$-approximation. We introduce a new linear programming relaxation of the problem, and a number of new techniques, that we hope will be helpful in designing more powerful algorithms for this and related problems.

Categories and Subject Descriptors

F.2.2 [**Nonnumerical Algorithms and Problems**]: Routing and layout

General Terms

Algorithms, Theory

Keywords

Approximation algorithms; routing problems; Node-Disjoint Paths

1. INTRODUCTION

In the Node-Disjoint Paths (NDP) problem, we are given an n-vertex graph G, and a collection $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of pairs of vertices of G, called *source-destination*, or *demand*, pairs. The goal is to route as many of the demand pairs as possible, by connecting each routed pair with a path, so that the resulting paths are node-disjoint. We denote by NDP-Planar the special case of the problem where the input graph G is planar, and by NDP-Grid the special case where G is the $(\sqrt{n} \times \sqrt{n})$-grid. NDP is one of the most basic problems in the area of graph routing, and it was initially introduced to the area in the context of VLSI design. In addition to being extensively studied in the area of approximation algorithms, this problem has played a central role in Robertson and Seymour's Graph Minor series. When the number of the demand pairs k is bounded by a constant, Robertson and Seymour [27, 29] have shown an efficient algorithm for the problem, as part of the series. When k is a part of input, the problem becomes NP-hard [15, 14], even in planar graphs [22], and even in grid graphs [21]. Despite the importance of this problem and many efforts, its approximability is still poorly understood. The following simple greedy algorithm achieves an $O(\sqrt{n})$-approximation [20]: while G contains any path connecting any demand pair, choose the shortest such path P, add P to the solution, and delete all vertices of P from G. Surprisingly, this elementary algorithm is the best currently known approximation algorithm for NDP, even for planar graphs. Until recently, this was also the best approximation algorithm for NDP-Grid. On the negative side, it is known that there is no $O(\log^{1/2-\delta} n)$-approximation algorithm for NDP for any constant δ, unless NP $\subseteq$ ZPTIME($n^{\text{poly}\log n}$) [4, 3]. To the best of our knowledge, the best negative result for NDP-Planar and for

*A full version of this paper is available at `http://arxiv.org/abs/1603.05520`

†Supported in part by NSF grant CCF-1318242.

‡Supported in part by NSF grant CCF-1318242.

§Part of the work done while the author was at the Toyota Technological Institute at Chicago.

STOC'16, June 19–21, 2016, Cambridge, MA, USA
© 2016 ACM. 978-1-4503-4132-5/16/06...$15.00
http://dx.doi.org/10.1145/2897518.2897538

NDP-Grid is APX-hardness [12]. Perhaps the biggest obstacle to obtaining better upper bounds on the approximability of NDP is that the common approach to designing approximation algorithms for this type of problems is to use the multicommodity flow relaxation, where instead of connecting the demand pairs with paths, we send a (possibly fractional) multicommodity flow between them. The integrality gap of this relaxation is known to be $\Omega(\sqrt{n})$, even for planar graphs, and even for grid graphs. In a recent work, Chuzhoy and Kim [12] showed an $\tilde{O}(n^{1/4})$-approximation algorithm for NDP-Grid, thus bypassing the integrality gap obstacle for this restricted family of graphs. The main result of this paper is an $\tilde{O}(n^{9/19})$-approximation algorithm for NDP-Planar. We also show that, if the value of the optimal solution to the NDP-Planar instance is OPT, then we can efficiently route $\Omega\left(\frac{\mathsf{OPT}^{1/19}}{\mathrm{poly}\log n}\right)$ demand pairs. Our algorithm is motivated by the work of [12] on NDP-Grid, and it relies on approximation algorithms for the NDP problem on a disc and on a cylinder, that we discuss next.

We start with the NDP problem on a disc, that we denote by NDP-Disc. In this problem, we are given a planar graph G, together with a set $\mathcal{M}$ of demand pairs as before, but we now assume that G can be drawn in a disc, so that all vertices participating in the demand pairs lie on its boundary. The NDP problem on a cylinder, NDP-Cylinder, is defined similarly, except that now we assume that we are given a cylinder Σ, obtained from the sphere, by removing two disjoint open discs (caps) from it. We denote the boundaries of the discs by Γ_1 and Γ_2 respectively. We assume that G can be drawn on Σ, so that all source vertices participating in the demand pairs in $\mathcal{M}$ lie on Γ_1, and all destination vertices lie on Γ_2. Robertson and Seymour [28] showed an algorithm, that, given an instance of the NDP-Disc or the NDP-Cylinder problem, decides whether all demand pairs in $\mathcal{M}$ can be routed simultaneously via node-disjoint paths, and if so, finds the routing efficiently. Moreover, for each of the two problems, they give an exact characterization of instances for which all pairs in $\mathcal{M}$ can be routed in G. Several other very efficient algorithms are known for both problems [26, 31]. However, we need to consider the optimization version of both problems, where we are no longer guaranteed that all demand pairs in $\mathcal{M}$ can be routed, and would like to route the largest possible subset of the demand pairs. We are not aware of any results for these two special cases of the NDP problem. In this paper, we provide $O(\log k)$-approximation algorithms for both problems.

Other Related Work. A problem closely related to NDP is the Edge-Disjoint Paths (EDP) problem. It is defined similarly, except that now the paths chosen to the solution are allowed to share vertices, and are only required to be edge-disjoint. It is easy to show, by using a line graph of the EDP instance, that NDP is more general than EDP (though this transformation inflates the number of the graph vertices, so it may not preserve approximation factors that depend on n). This relationship breaks down in planar graphs, since the resulting NDP instance may no longer be planar. The approximability status of EDP is very similar to that of NDP: there is an $O(\sqrt{n})$-approximation algorithm [10], and it is known that there is no $O(\log^{1/2-\delta} n)$-approximation algorithm for any constant δ, unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\mathrm{poly}\log n})$ [4, 3]. We do not know whether our techniques can be used to obtain improved approximation algorithms for EDP in planar graphs. As in the NDP problem, we can use the standard multicommodity flow LP-relaxation of the problem, in order to obtain an $O(\sqrt{n})$-approximation algorithm, and the integrality gap of the LP-relaxation is $\Omega(\sqrt{n})$ even in planar graphs. For several special cases of the problem better algorithms are known: Kleinberg [17], building on the work of Chekuri, Khanna and Shepherd [9, 8], has shown an $O(\log^2 n)$-approximation LP-rounding algorithm for even-degree planar graphs. Aumann and Rabani [5] showed an $O(\log^2 n)$-approximation algorithm for EDP on grid graphs, and Kleinberg and Tardos [19, 18] showed $O(\log n)$-approximation algorithms for broader classes of nearly-Eulerian uniformly high-diameter planar graphs, and nearly-Eulerian densely embedded graphs. Recently, Kawarabayashi and Kobayashi [16] showed a factor $O(\log n)$-approximation algorithm for EDP when the input graph is either 4-edge-connected planar or Eulerian planar. It appears that the restriction of the graph G to be Eulerian, or nearly-Eulerian, makes the EDP problem significantly simpler, and in particular improves the integrality gap of the LP-relaxation. The analogue of the grid graph for the EDP problem is the wall graph: the integrality gap of the standard LP-relaxation for EDP on wall graphs is $\Omega(\sqrt{n})$, and until recently, no better than $O(\sqrt{n})$-approximation algorithms for EDP on walls were known. The work of [12] gives an $\tilde{O}(n^{1/4})$-approximation algorithm for EDP on wall graphs.

A variation of the NDP and EDP problems, where small congestion is allowed, has been a subject of extensive study. In the NDP with congestion (NDPwC) problem, the input is the same as in the NDP problem, and we are additionally given a non-negative integer c. The goal is to route as many of the demand pairs as possible with congestion at most c: that is, every vertex may participate in at most c paths in the solution. EDP with Congestion (EDPwC) is defined similarly, except that now the congestion bound is imposed on edges and not vertices. The classical randomized rounding technique of Raghavan and Thompson [24] gives a constant-factor approximation for both problems, if the congestion c is allowed to be as high as $\Theta(\log n/\log\log n)$. A recent line of work [9, 23, 2, 25, 11, 13, 7, 6] has lead to an $O(\mathrm{poly}\log k)$-approximation for both NDPwC and EDPwC problems, with congestion $c = 2$. In planar graphs, a constant-factor approximation with congestion 2 is known for EDP [30]. All these algorithms perform LP-rounding of the standard multicommodity flow LP-relaxation of the problem and so it is unlikely that they can be extended to routing with no congestion.

Our Results and Techniques. Given an instance $(G, \mathcal{M})$ of the NDP problem, we denote by $\mathsf{OPT}(G, \mathcal{M})$ the value of the optimal solution to it. Our first result is an approximation algorithm for NDP-Disc and NDP-Cylinder.

Theorem 1.1 *There is an efficient $O(\log k)$-approximation algorithm for the* NDP-Disc *and the* NDP-Cylinder *problems, where k is the number of the demand pairs in the instance.*

The main result of our paper is summarized in the following two theorems.

Theorem 1.2 *There is an efficient $O(n^{9/19} \cdot \operatorname{poly}\log n)$-approximation algorithm for the* NDP-Planar *problem.*

Theorem 1.3 *There is an efficient algorithm, that, given an instance $(G, \mathcal{M})$ of* NDP-Planar*, computes a routing of $\Omega\left(\frac{(\mathsf{OPT}(G,\mathcal{M}))^{1/19}}{\operatorname{poly}\log n}\right)$ demand pairs of $\mathcal{M}$ via node-disjoint paths in G.*

Notice that when $\mathsf{OPT}(G, \mathcal{M})$ is small, Theorem 1.3 gives a much better that $\tilde{O}(n^{9/19})$-approximation.

We now give a high-level intuitive overview of the proof of Theorem 1.2. Given an instance $(G, \mathcal{M})$ of the NDP problem, we denote by $\mathcal{T}$ the set of vertices participating in the demand pairs in $\mathcal{M}$, and we refer to them as *terminals*. We start with a quick overview of the $\tilde{O}(n^{1/4})$-approximation algorithm of [12] for the NDP-Grid problem, since their algorithm was the motivation for this work. The main observation of [12] is that the instances of NDP-Grid, for which the multicommodity flow relaxation exhibits the $\Omega(\sqrt{n})$ integrality gap, have terminals close to the grid boundary. When all terminals are at a distance of at least $\Omega(n^{1/4})$ from the boundary of the grid, one can find an $\tilde{O}(n^{1/4})$-approximation via LP-rounding (but unfortunately the integrality gap remains polynomial in n even in this case). When the terminals are close to the grid boundary, the integrality gap of the LP-relaxation becomes $\Omega(\sqrt{n})$. However, this special case of NDP-Grid can be easily approximated via simple dynamic programming; we omit the details here. Overall, we partition the demand pairs of $\mathcal{M}$ into two subsets, depending on whether the terminals lie close to or far from the grid boundary, and obtain an $\tilde{O}(n^{1/4})$-approximation for each of the two resulting problem instances separately, selecting the better of the two solutions as our output.

This idea is much more difficult to implement in general planar graphs. For one thing, the notion of the "boundary" of a planar graph is meaningless - any face in the drawing of the planar graph can be chosen as the outer face. We note that the standard multicommodity flow LP-relaxation performs poorly not only when all terminals are close to the boundary of a single face (a case somewhat similar to NDP-Disc), but also when there are two faces F and F', and for every demand pair $(s,t) \in \mathcal{M}$, s is close to the boundary of F and t is close to the boundary of F' (this setting is somewhat similar to NDP-Cylinder). The notion of "distance", when deciding whether the terminals lie close to or far from a face boundary is also not well-defined, since we can subdivide edges and artificially modify the graph in various ways in order to manipulate the distances without significantly affecting routings. Intuitively, we would like to define the distances between the terminals in such a way that, on the one hand, whenever we find a set $\mathcal{M}' \subseteq \mathcal{M}$ of demand pairs, such that all terminals participating in the pairs in $\mathcal{M}'$ are far enough from each other, then we can route a large subset of the demand pairs in $\mathcal{M}'$. On the other hand, if we find a set $\mathcal{M}'' \subseteq \mathcal{M}$ of demand pairs, with all terminals participating in the pairs in $\mathcal{M}''$ being close to the boundary of some face (or a pair of faces), then we can find a good approximate solution to instance $(G, \mathcal{M}'')$ (for example, by reducing the problem to NDP-Disc or NDP-Cylinder). Since we do not know beforehand which face (or faces) will be chosen as the "boundary" of the graph, we cannot partition the problem into two sub-problems and employ different techniques to solve each sub-problem as we did for NDP-Grid. Instead, we need a single framework in which both cases can be handled.

We assume that every terminal participates in exactly one demand pair, and that the degree of every terminal is 1. This can be done via a standard transformation, where we create several copies of each terminal, and connect them to the original terminal. This transformation may introduce up to $O(n^2)$ new vertices. Since we are interested in obtaining an $\tilde{O}(n^{9/19})$-approximation for NDP-Planar, we denote by N the number of the non-terminal vertices in the new graph G. Abusing the notation, we denote the total number of vertices in the new problem instance by n. It is now enough to obtain an $\tilde{O}(N^{9/19})$-approximation for the new problem instance.

Our first step is to define a new LP-relaxation of the problem. We assume that we have guessed correctly the value OPT of the optimal solution. We start with the standard multicommodity flow LP-relaxation, where we try to send OPT flow units between the demand pairs, so that the maximum amount of flow through any vertex is bounded by 1. We then add the following new set of constraints to the LP: for every subset $\mathcal{M}' \subseteq \mathcal{M}$ of the demand pairs, for every integer $\mathsf{OPT}(G, \mathcal{M}') \leq z \leq k$, the total amount of flow routed between the demand pairs in $\mathcal{M}'$ is no more than z. Adding this type of constraints may seem counter-intuitive. We effectively require that the LP solves the problem exactly, and naturally we cannot expect to be able to do so efficiently. Since the number of the resulting constraints is exponential in k, and since we do not know the values $\mathsf{OPT}(G, \mathcal{M}')$, we indeed cannot solve this LP efficiently. In fact, our algorithm does not attempt to solve the LP exactly. Instead, we employ the Ellipsoid algorithm, that in every iteration produces a potential solution to the LP-relaxation. We then show an algorithm that, given such a potential solution, either finds an integral solution routing $\tilde{\Omega}(\mathsf{OPT}/N^{9/19})$ demand pairs, or it returns some subset $\mathcal{M}' \subseteq \mathcal{M}$ of demand pairs, whose corresponding LP-constraint is violated. Therefore, we use our approximation algorithm as a separation oracle for the Ellipsoid algorithm. We are then guaranteed that after $\operatorname{poly}(n)$ iterations, we will obtain a solution routing the desired number of demand pairs, as only $\operatorname{poly}(n)$ iterations are required for the Ellipsoid algorithm in order to find a feasible LP-solution.

The heart of the proof of Theorem 1.2 is then an algorithm that, given a potential (possibly infeasible) solution to the LP-relaxation, either finds an integral solution routing $\tilde{\Omega}(\mathsf{OPT}/N^{9/19})$ demand pairs, or returns some subset $\mathcal{M}' \subseteq \mathcal{M}$ of demand pairs, whose corresponding LP-constraint is violated. We can assume without loss of generality that the fractional solution we are given satisfies all the standard multicommodity flow constraints, as this can be verified efficiently. For simplicity of exposition, we assume that every demand pair in $\mathcal{M}$ sends the same amount of w^* flow units to each other.

We assume for now that the set $\mathcal{T}$ of terminals is α_{WL}-well-linked, for $\alpha_{\text{WL}} = \Theta(w^*/\log n)$ - that is, for every pair $(\mathcal{T}', \mathcal{T}'')$ of disjoint equal-sized subsets of vertices of $\mathcal{T}$, we can connect vertices of $\mathcal{T}'$ to vertices of $\mathcal{T}''$ by at least $\alpha_{\text{WL}} \cdot |\mathcal{T}'|$ node-disjoint paths. We discuss this assumption in more detail below. We assume that we are given a drawing of G on the sphere. Our first step is to define the notion of distances between the terminals. In order to do so, we first construct *enclosures* around them. Throughout the proof,

we use a parameter $\Delta = \mathsf{OPT}^{2/19}$. We say that a curve γ on the sphere is a *G-normal curve* iff it intersects the drawing of G only at its vertices. The length of such a curve is the number of vertices of G it contains. An enclosure around a terminal t is a disc D_t containing t, whose boundary, that we denote by C_t, is a G-normal curve of length exactly Δ, so that at most $O(\Delta/\alpha_{\mathrm{WL}})$ terminals lie in D_t. We show an efficient algorithm to construct the enclosures D_t around the terminals t, so that the following additional conditions hold: (i) if $D_t \subseteq D_{t'}$ for any pair $t, t' \in \mathcal{T}$ of terminals, then $D_t = D_{t'}$; and (ii) if $D_t \cap D_{t'} = \emptyset$, then there are Δ node-disjoint paths connecting the vertices of C_t to the vertices of $C_{t'}$. We then define the distances between pairs of terminals: for every pair (t, t') of terminals, distance $d(t, t')$ is the length of the shortest G-normal curve, connecting a vertex of C_t to a vertex of $C_{t'}$.

Next, we show that one of the following has to happen: either there is a large collection $\tilde{\mathcal{M}} \subseteq \mathcal{M}$ of demand pairs, such that all terminals participating in the pairs in $\tilde{\mathcal{M}}$ are at a distance at least $\Omega(\Delta)$ from each other; or there is a large collection $\tilde{\mathcal{M}}' \subseteq \mathcal{M}$ of demand pairs, and two faces F, F' in the drawing of G (with possibly $F = F'$), such that for every demand pair in $\tilde{\mathcal{M}}'$, one of its terminals is within distance at most $\tilde{O}(\Delta)$ from the boundary of F, and the other is within distance at most $\tilde{O}(\Delta)$ from the boundary of F'. In the former case, we show that we can route a large subset of the demand pairs in $\tilde{\mathcal{M}}$ via node-disjoint paths, by constructing a special routing structure called a crossbar (this construction exploits the well-linkedness of the terminals and the paths connecting the enclosures). In the latter case, we reduce the problem to NDP-Disc or NDP-Cylinder, depending on the distance between the faces F and F', and employ the approximation algorithms for these problems to route $\tilde{\Omega}\left(\frac{\mathsf{OPT}(G,\tilde{\mathcal{M}}')}{\mathrm{poly}(\Delta)}\right)$ demand pairs from $\tilde{\mathcal{M}}'$ in G. If the resulting number of demand pairs routed is close enough to OPT, then we return this as our final solution. Otherwise, we show that the LP-constraint corresponding to the set $\tilde{\mathcal{M}}'$ of demand pairs is violated, or equivalently, the amount of flow sent by the LP solution between the demand pairs in $\tilde{\mathcal{M}}'$ is greater than $\mathsf{OPT}(G, \tilde{\mathcal{M}}')$.

So far we have assumed that the terminals participating in the demand pairs in $\mathcal{M}$ are α_{WL}-well-linked. In general this may not be the case. Using standard techniques, we can perform a well-linked decomposition: that is, compute a subset $U \subseteq V(G)$ of at most $\mathsf{OPT}/64$ vertices, such that, if we denote the set of all connected components of $G \setminus U$ by $\{G_1, \ldots, G_r\}$, and for each $1 \leq i \leq r$, we denote by $\mathcal{M}_i \subseteq G_i$ the set of the demand pairs contained in G_i, then the terminals participating in the demand pairs in $\mathcal{M}_i$ are α_{WL}-well-linked in G_i. We are then guaranteed that $\sum_{i=1}^{r} \mathsf{OPT}(G_i, \mathcal{M}_i) \geq \frac{63}{64}\mathsf{OPT}$. It is then tempting to apply the algorithm described above to each of the graphs separately. Indeed, if, for each $1 \leq i \leq r$, we find a set $\mathcal{P}_i$ of node-disjoint paths, routing $\Omega\left(\frac{\mathsf{OPT}(G_i,\mathcal{M}_i)}{N_i^{9/19}\cdot \mathrm{poly}\log n}\right)$ demand pairs of $\mathcal{M}_i$ in G_i (where N_i denotes the number of the non-terminal vertices in G_i), then we obtain an $O(N^{9/19} \cdot \mathrm{poly}\log n)$-approximate solution overall. Assume now that for some $1 \leq i \leq r$, we find a subset $\mathcal{M}_i' \subseteq \mathcal{M}_i$ of demand pairs, such that $\mathsf{OPT}(G_i, \mathcal{M}_i') < w^*|\mathcal{M}_i'|/16$. Unfortunately, the set $\mathcal{M}_i'$ of demand pairs does not necessarily define a violated LP-constraint, since it is possible that $\mathsf{OPT}(G, \mathcal{M}_i') >> \mathsf{OPT}(G_i, \mathcal{M}_i')$, if the optimal routing uses many vertices of U (and possibly from some other graphs G_j). In general, the number of vertices in set U is relatively small compared to OPT, so in the global accounting across all instances $(G_{i'}, \mathcal{M}_{i'})$, only a small number of paths can use the vertices of U. But for a specific instance $(G_i, \mathcal{M}_i)$, it is possible that most paths in the optimal solution to instance $(G, \mathcal{M}_i)$ use the vertices of U. In order to overcome this difficulty, we need to perform a careful global accounting across all resulting instances $(G_{i'}, \mathcal{M}_{i'})$.

Organization. We start with preliminaries in Section 2. Section 3 is devoted to the proof of Theorem 1.1. Due to lack of space, we only provide a brief sketch of the proof. Sections 4–7 are devoted to the proof of Theorem 1.2: Section 4 provides an overview of the algorithm and some initial steps; Section 5 introduces the main technical tools that we use: enclosures, shells, and a partition of the terminals into subsets; and Sections 6 and 7 deal with Case 1 (when many terminals are far from each other) and Case 2 (when many terminals are close to the boundaries of at most two faces), respectively. We prove Theorem 1.3 in Section 8. All proofs omitted from this extended abstract can be found in the full version of the paper available on Arxiv.

2. PRELIMINARIES

Given any graph G and a set $\mathcal{M}$ of demand pairs in G, for any subset $\mathcal{M}' \subseteq \mathcal{M}$ of the demand pairs, we denote by $\mathcal{T}(\mathcal{M}')$ the set of all vertices participating in the demand pairs in $\mathcal{M}'$. For any subset $\mathcal{M}' \subseteq \mathcal{M}$ of the demand pairs, and any sub-graph $H \subseteq G$, let $\mathsf{OPT}(H, \mathcal{M}')$ denote the value of the optimal solution to instance $(H, \mathcal{M}')$.

Given a drawing of any planar graph H in the plane, and given any cycle C in H, we denote by $D(C)$ the unique disc in the plane whose boundary is C. Similarly, if C is a closed simple curve in the plane, $D(C)$ is the unique disc whose boundary is C. When the graph H is drawn on the sphere, there are two discs whose boundaries are C. In such cases we will explicitly specify which of the two discs we refer to. Given any disc D (in the plane or on the sphere), we use D° to denote the disc D without its boundary. We say that a vertex of H belongs to disc D, and denote $v \in D$, if v is drawn inside D or on its boundary. Given a planar graph G, drawn on a surface Σ, we say that a curve C in Σ is *G-normal*, iff it intersects the drawing of G at vertices only. The set of vertices of G lying on C is denoted by $V(C)$, and the length of C is $\ell(C) = |V(C)|$. For any disc D, whose boundary is a G-normal curve, we denote by $V(D)$ the set of all vertices of G lying inside D or on its boundary.

Sparsest Cut. In this paper we use the node version of the sparsest cut problem, defined as follows. Suppose we are given a graph $G = (V, E)$ with a subset $\mathcal{T} \subseteq V$ of its vertices called terminals. A vertex cut is a tri-partition (A, C, B) of V, such that there are no edges in G with one endpoint in A and the other in B. If $(A \cup C) \cap \mathcal{T}, (B \cup C) \cap \mathcal{T} \neq \emptyset$, then the sparsity of the cut (A, C, B) is $\frac{|C|}{\min\{|A\cap\mathcal{T}|,|B\cap\mathcal{T}|\}+|C\cap\mathcal{T}|}$. The sparsest cut in G with respect to the set $\mathcal{T}$ of terminals is a vertex cut (A, C, B) with $(A \cup C) \cap \mathcal{T}, (B \cup C) \cap \mathcal{T} \neq \emptyset$, whose sparsity is the smallest among all such cuts. Amir, Krauthgamer and Rao [1] showed an efficient algorithm, that, given any planar graph G with a set $\mathcal{T} \subseteq V(G)$ of ter-

minal vertices, computes a vertex cut (A, C, B) in G, whose sparsity with respect to $\mathcal{T}$ is within a constant factor of the optimal one. We denote this algorithm by $\mathcal{A}_{\text{AKR}}$, and the approximation factor it achieves by α_{AKR}, so α_{AKR} is an absolute constant.

Tight Concentric Cycles. We start with the following definition.

Definition 2.1 *Given a planar graph H drawn in the plane and a vertex $v \in V(H)$ that is not incident to the infinite face,* min-cycle(H, v) *is the cycle C in H, such that: (i) $v \in D^{\circ}(C)$; and (ii) among all cycles satisfying (i), C is the one for which $D(C)$ is minimal inclusion-wise.*

It is easy to see that min-cycle(H, v) is uniquely defined. Indeed, consider the graph $H \setminus v$, and the face F in the drawing of $H \setminus v$ where v used to reside. Then the boundary of F contains exactly one cycle C with $D(C)$ containing v, and $C =$ min-cycle(H, v). We next define a family of tight concentric cycles.

Definition 2.2 *Suppose we are given a planar graph H, an embedding of H in the plane, a simple closed H-normal curve C, and an integral parameter $r \geq 1$. A family of r tight concentric cycles around C is a sequence $Z_1, Z_2, \ldots, Z_r$ of disjoint simple cycles in H, with the following properties:*

- $D(C) \subsetneq D(Z_1) \subsetneq D(Z_2) \subsetneq \cdots \subsetneq D(Z_r)$;
- *if H' is the graph obtained from H by contracting all vertices lying in $D(C)$ into a super-node a, then $Z_1 =$* min-cycle(H', a)*; and*
- *for every $1 < h \leq r$, if H' is the graph obtained from H by contracting all vertices lying in $D(Z_{h-1})$ into a super-node a, then $Z_h =$* min-cycle(H', a).

3. ROUTING ON A DISC AND ON A CYLINDER

In this section we prove Theorem 1.1. In order to do so, we define a new problem, called Demand Pair Selection Problem (DPSP), and show an 8-approximation algorithm for it. We then show that both NDP-Disc and NDP-Cylinder reduce to DPSP.

Demand Pair Selection Problem. We assume that we are given two disjoint directed paths, σ and σ', and a collection $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of pairs of vertices of $\sigma \cup \sigma'$ that are called demand pairs, where all vertices of $S = \{s_1, \ldots, s_k\}$ lie on σ, and all vertices of $T = \{t_1, \ldots, t_k\}$ lie on σ' (not necessarily in this order). We refer to the vertices of S and T as the *source* and the *destination* vertices, respectively. Note that the same vertex of σ may participate in several demand pairs, and the same is true for the vertices of σ'. Given any pair a, a' of vertices of σ, with a lying before a' on σ, we sometimes denote by (a, a') the sub-path of σ between a and a' (that includes both these vertices). We define sub-paths of σ' similarly.
For every pair $v, v' \in V(\sigma)$ of vertices, we denote $v \prec v'$ if v lies strictly before v' on σ, and we denote $v \preceq v'$, if $v \prec v'$ or $v = v'$ hold. Similarly, for every pair $v, v' \in V(\sigma')$ of vertices, we denote $v \prec v'$ if v lies strictly before v' on σ', and we denote $v \preceq v'$, if $v \prec v'$ or $v = v'$ hold. We need the following definitions.

Definition 3.1 *Suppose we are given two pairs (a, b) and (a', b') of vertices of $\sigma \cup \sigma'$, with $a, a' \in \sigma$ and $b, b' \in \sigma'$. We say that (a, b) and (a', b')* cross *iff one of the following holds: either (i) $a = a'$; or (ii) $b = b'$; or (iii) $a \prec a'$ and $b' \prec b$; or (iv) $a' \prec a$ and $b \prec b'$.*

Definition 3.2 *We say that a subset $\mathcal{M}' \subseteq \mathcal{M}$ of demand pairs is* non-crossing *iff for all distinct pairs $(s, t), (s', t') \in \mathcal{M}'$, (s, t) and (s', t') do not cross.*

Our goal is to select the largest-cardinality non-crossing subset $\mathcal{M}' \subseteq \mathcal{M}$ of demand pairs, satisfying a collection $\mathcal{K}$ of constraints. Set $\mathcal{K}$ of constraints is given as part of the problem input, and consists of four subsets, $\mathcal{K}_1, \ldots, \mathcal{K}_4$, where constraints in set $\mathcal{K}_i$ are called *type-i constraints*. Every constraint $K \in \mathcal{K}$ is specified by a quadruple (i, a, b, w), where $i \in \{1, 2, 3, 4\}$ is the constraint type, $a, b \in V(\sigma \cup \sigma')$, and $1 \leq w \leq |\mathcal{M}|$ is an integer.
For every type-1 constraint $K = (1, a, b, w) \in \mathcal{K}_1$, we have $a, b \in V(\sigma)$ with $a \prec b$. The constraint is associated with the sub-path $I = (a, b)$ of σ. We say that a subset $\mathcal{M}' \subseteq \mathcal{M}$ of demand pairs *satisfies* K iff the total number of the source vertices participating in the demand pairs of $\mathcal{M}'$ that lie on I is at most w.
Similarly, for every type-2 constraint $K = (2, a, b, w) \in \mathcal{K}_2$, we have $a, b \in V(\sigma')$ with $a \prec b$, and the constraint is associated with the sub-path $I = (a, b)$ of σ'. A set $\mathcal{M}' \subseteq \mathcal{M}$ of demand pairs satisfies K iff the total number of the destination vertices participating in the demand pairs in $\mathcal{M}'$ that lie on I is at most w.
For each type-3 constraint $K = (3, a, b, w) \in \mathcal{K}_3$, we have $a \in V(\sigma)$ and $b \in V(\sigma')$. The constraint is associated with the sub-path L_a of σ between the first vertex of σ and a (including both these vertices), and the sub-path R_b of σ' between b and the last vertex of σ' (including both these vertices). We say that a demand pair $(s, t) \in \mathcal{M}$ *crosses* K iff $s \in L_a$ and $t \in R_b$. A set $\mathcal{M}' \subseteq \mathcal{M}$ of demand pairs satisfies K iff the total number of pairs $(s, t) \in \mathcal{M}'$ that cross K is bounded by w.
Finally, for each type-4 constraint $K = (4, a, b, w) \in \mathcal{K}_4$, we also have $a \in V(\sigma)$ and $b \in V(\sigma')$. The constraint is associated with the sub-path R_a of σ between a and the last vertex of σ (including both these vertices), and the sub-path L_b of σ' between the first vertex of σ' and b (including both these vertices). We say that a demand pair $(s, t) \in \mathcal{M}$ crosses K iff $s \in R_a$ and $t \in L_b$. A set $\mathcal{M}' \subseteq \mathcal{M}$ of demand pairs satisfies K iff the total number of pairs $(s, t) \in \mathcal{M}'$ that cross K is bounded by w.
Given the paths σ, σ', the set $\mathcal{M}$ of the demand pairs, and the set $\mathcal{K}$ of constraints as above, the goal in the DPSP problem is to select a maximum-cardinality non-crossing subset $\mathcal{M}' \subseteq \mathcal{M}$ of demand pairs, such that all constraints in $\mathcal{K}$ are satisfied by $\mathcal{M}'$. The proof of the following theorem relies on dynamic programming, and is omitted from this extended abstract.

Theorem 3.1 *There is an efficient 8-approximation algorithm for* DPSP.

We use Theorem 3.1 in order to design efficient algorithms for NDP-Disc and NDP-Cylinder, achieving a factor $O(\log k)$-approximation for both problems, by reducing them to DPSP with an $O(\log k)$ loss in the approximation factor. The reduction from NDP-Disc to DPSP uses the characterization of routable instances of NDP-Disc due to Robertson and Seymour [28]. We then reduce NDP-Cylinder to NDP-Disc and DPSP directly. The remainder of the proof is omitted from this extended abstract.

4. ALGORITHM SETUP

The rest of this paper mostly focuses on proving Theorem 1.2; we prove Theorem 1.3 using the techniques we employ for the proof of Theorem 1.2 in Section 8.

We assume without loss of generality that the input graph G is connected - otherwise we can solve the problem separately on each connected component of G. Let $\mathcal{T} = \mathcal{T}(\mathcal{M})$. It is convenient for us to assume that every terminal participates in exactly one demand pair, and that the degree of every terminal is 1. This can be achieved via a standard transformation of the input instance, where we add a new collection of terminals, connecting them to the original terminals. This transformation preserves planarity, but unfortunately it can increase the number of the graph vertices. If the original graph G contained n vertices, then $|\mathcal{M}|$ can be as large as n^2, and so the new graph may contain up to n^2+n vertices, while our goal is to obtain an $\tilde{O}(n^{9/19})$-approximation. In order to overcome this difficulty, we denote by N the number of the non-terminal vertices in the new graph G, so N is bounded by the total number of vertices in the original graph, and by n the total number of all vertices in the new graph, so $n = O(N^2)$. Our goal is then to obtain an efficient $O(N^{9/19} \cdot \operatorname{poly}\log n)$-approximation for the new problem instance. From now on we assume that every terminal participates in exactly one demand pair, and the degree of every terminal is 1. We denote $|\mathcal{M}| = k$. Throughout the algorithm, we define a number of sub-instances of the instance $(G, \mathcal{M})$, but we always use k to denote the number of the demand pairs in this initial instance. We can assume that $k > 100$, as otherwise we can return a routing of a single demand pair.

We assume that we are given a drawing of G on the sphere. Throughout the algorithm, we will sometimes select some face of G as the outer face, and consider the resulting planar drawing of G.

4.1 LP-Relaxations

Let us start with the standard multicommodity flow LP-relaxation of the problem. Let G' be the directed graph, obtained from G by bi-directing its edges. For every edge $e \in E(G')$, for each $1 \leq i \leq k$, there is an LP-variable $f_i(e)$, whose value is the amount of the commodity-i flow through edge e. We denote by x_i the total amount of commodity-i flow sent from s_i to t_i. For every vertex v, let $\delta^+(v)$ and $\delta^-(v)$ denote the sets of its out-going and in-coming edges, respectively. We denote $[k] = \{1, \ldots, k\}$. The standard LP-relaxation of the NDP problem, that we denote by (LP-flow1), appears in Figure 1.

We make two changes to (LP-flow1). First, we assume that we know the value X^* of the optimal solution, and instead of the objective function, we add the constraint $\sum_{i=1}^{k} x_i \geq X^*$. We can do so using standard methods, by repeatedly guessing the value X^* and running the algorithm for each such value. It is enough to show that the algorithm routes $\Omega\left(\frac{X^*}{N^{9/19} \cdot \operatorname{poly}\log n}\right)$ demand pairs, when the value X^* is guessed correctly.

Recall that for a subset $\mathcal{M}' \subseteq \mathcal{M}$ of the demand pairs, and a sub-graph $H \subseteq G$, $\mathsf{OPT}(H, \mathcal{M}')$ denotes the value of the optimal solution to instance $(H, \mathcal{M}')$. For every subset $\mathcal{M}' \subseteq \mathcal{M}$ of the demand pairs, we add the constraint that the total flow between all pairs in $\mathcal{M}'$ is no more than z, for all integers z between $\mathsf{OPT}(G, \mathcal{M}')$ and k. We now obtain a linear program, denoted by (LP-flow2), that has no objective function, so we are only interested in finding a feasible solution (see Figure 2).

We say that a solution to (LP-flow2) is *semi-feasible* iff all constraints of types (1)–(4) and (6) are satisfied. Notice that the number of the constraints in (LP-flow2) is exponential in k. In order to solve it, we use the Ellipsoid Algorithm with a separation oracle, where our approximation algorithm itself will serve as the separation oracle. This is done via the following theorem, which is our main technical result.

Theorem 4.1 *There is an efficient algorithm, that, given a semi-feasible solution to (LP-flow2), either computes a routing of at least $\Omega\left(\frac{X^*}{N^{9/19} \cdot \operatorname{poly}\log n}\right)$ demand pairs of $\mathcal{M}$ via node-disjoint paths, or returns a constraint of type (5), that is violated by the current solution.*

We can now obtain an $O\left(N^{9/19} \cdot \operatorname{poly}\log n\right)$-approximation algorithm for NDP-Planar via the Ellipsoid algorithm. In every iteration, we start with some semi-feasible solution to (LP-flow2), and apply the algorithm from Theorem 4.1 to it. If the outcome is a solution routing at least $\Omega\left(\frac{X^*}{N^{9/19} \cdot \operatorname{poly}\log n}\right)$ demand pairs in $\mathcal{M}$, then we obtain the desired approximate solution to the problem, assuming that X^* was guessed correctly. Otherwise, we obtain a violated constraint of type (5), and continue to the next iteration of the Ellipsoid Algorithm. Since the Ellipsoid Algorithm is guaranteed to terminate with a feasible solution after $\operatorname{poly}(n)$ iterations, this gives an efficient algorithm that is guaranteed to return a solution of value $\Omega\left(\frac{X^*}{N^{9/19} \cdot \operatorname{poly}\log n}\right)$. From now on we focus on proving Theorem 4.1.

We now assume that we are given some semi-feasible solution (x, f) to (LP-flow2), and define a new fractional solution based on it, where the flow between every demand pair is either 0 or w^*, for some value $w^* > 0$. First, for each demand pair (s_i, t_i) with $x_i \leq \frac{1}{2k}$, we set $x_i = 0$ and we set the corresponding flow values $f_i(e)$ for all edges $e \in E(G')$ to 0. Since we can assume that $X^* \geq 1$ if the graph is connected, the total amount of flow between the demand pairs remains at least $X^*/2$. We then partition the remaining demand pairs into $q = \lceil \log 2k \rceil$ subsets, where for $1 \leq j \leq q$, set $\mathcal{M}_j$ contains all demand pairs (s_i, t_i) with $\frac{1}{2^j} < x_i \leq \frac{1}{2^{j-1}}$. There is some index $1 \leq j^* \leq q$, such that the total flow between the demand pairs in $\mathcal{M}_{j^*}$ is at least $\Omega(X^*/\log k)$. Let $w^* = \frac{1}{2^j}$. We further modify the LP-solution, as follows. First, for every demand pair $(s_i, t_i) \notin \mathcal{M}_{j^*}$, we set $x_i = 0$, and the corresponding flow values $f_i(e)$ for all edges $e \in E(G')$ to 0. Next, for every demand pair $(s_i, t_i) \in \mathcal{M}_{j^*}$, we let $\beta_i = w^*/x_i$, so $\beta_i \leq 1$. We set $x_i = w^*$, and the new flow values $f_i(e)$ are obtained by scaling the original values by factor β_i. This gives a new solution to (LP-flow2), that we denote by (x', f'). The total amount of flow sent

(LP-flow1)

$$\begin{aligned}
\max \quad & \sum_{i=1}^{k} x_i \\
\text{s.t.} \quad & \\
& \sum_{e\in\delta^+(s_i)} f_i(e) = x_i && \forall i \in [k] \\
& \sum_{e\in\delta^+(v)} f_i(e) = \sum_{e\in\delta^-(v)} f_i(e) && \forall i \in [k], \forall v \in V(G') \setminus \{s_i, t_i\} \qquad \text{(flow conservation)} \\
& \sum_{e\in\delta^+(v)} \sum_{i=1}^{k} f_i(e) \le 1 && \forall v \in V(G') \qquad \text{(vertex capacity constraints)} \\
& f_i(e) \ge 0 && \forall i \in [k], \forall e \in E(G')
\end{aligned}$$

Figure 1: Basic LP

(LP-flow2)

$$\begin{aligned}
& \sum_{i=1}^{k} x_i \ge X^* && && (1) \\
& \sum_{e\in\delta^+(s_i)} f_i(e) = x_i && \forall i \in [k] && (2) \\
& \sum_{e\in\delta^+(v)} f_i(e) = \sum_{e\in\delta^-(v)} f_i(e) && \forall i \in [k], \forall v \in V(G') \setminus \{s_i, t_i\} \quad \text{(flow conservation)} && (3) \\
& \sum_{e\in\delta^+(v)} \sum_{i=1}^{k} f_i(e) \le 1 && \forall v \in V(G') \qquad \text{(vertex capacity constraints)} && (4) \\
& \sum_{(s_i,t_i)\in\mathcal{M}'} x_i \le z && \forall \mathcal{M}' \subseteq \mathcal{M}, \forall z \in \mathbb{Z} : \mathsf{OPT}(G, \mathcal{M}') \le z \le k && (5) \\
& f_i(e) \ge 0 && \forall i \in [k], \forall e \in E(G') && (6)
\end{aligned}$$

Figure 2: Extended LP

in this solution is $\Omega(X^*/\log k)$, and it is easy to verify that constraints (2)–(4) and (6) are satisfied. For every demand pair $(s_i, t_i) \in \mathcal{M}_{j^*}$, $x'_i = w^*$, and for all other demand pairs (s_i, t_i), $x'_i = 0$. It is easy to see that for every demand pair (s_i, t_i), $x'_i \le x_i$. Therefore, if we find a constraint of type (5) that is violated by the new solution, then it is also violated by the old solution. Our goal now is to either find a feasible solution routing $\Omega\left(\frac{X^*}{N^{9/19}\cdot \mathrm{poly}\log n}\right)$ demand pairs, or to find a constraint of type (5) violated by the new solution. Since from now on we only focus on demand pairs in $\mathcal{M}_{j^*}$, for simplicity we denote $\mathcal{M} = \mathcal{M}_{j^*}$.

4.2 Well-Linked Decomposition

Like many other approximation algorithms for routing problems, we decompose our input instance into a collection of sub-instances that have some useful well-linkedness properties. Since the routing is on node-disjoint paths, we need to use a slightly less standard notion of node-well-linkedness, defined below. Throughout this paper, we use a parameter $\alpha_{\text{WL}} = w^*/(512 \cdot \alpha_{\text{AKR}} \cdot \log k)$.

Definition 4.1 *Given a graph H and a set $\mathcal{T}'$ of its vertices, we say that $\mathcal{T}'$ is α_{WL}-*well-linked *in H iff for every pair $\mathcal{T}_1, \mathcal{T}_2$ of disjoint equal-sized subsets of $\mathcal{T}$, there is a set $\mathcal{P}$ of at least $\alpha_{\text{WL}} \cdot |\mathcal{T}_1|$ node-disjoint paths in H, connecting vertices of $\mathcal{T}_1$ to vertices of $\mathcal{T}_2$.*

Definition 4.2 *Given a sub-graph $H \subseteq G$ and a subset $\mathcal{M}' \subseteq \mathcal{M}$ of demand pairs with $\mathcal{T}(\mathcal{M}') \subseteq V(H)$, we say that $(H, \mathcal{M}')$ is a* well-linked instance, *iff $\mathcal{T}(\mathcal{M}')$ is α_{WL}-well-linked in H.*

The following theorem uses standard techniques, and its proof is omitted from this extended abstract.

Theorem 4.2 *There is an efficient algorithm to compute a collection $G_1, \ldots, G_r$ of disjoint sub-graphs of G, and for each $1 \le j \le r$, a set $\mathcal{M}^j \subseteq \mathcal{M}$ of demand pairs with $\mathcal{T}(\mathcal{M}^j) \subseteq V(G_j)$, such that:*

- *For all $1 \le j \le r$, $(G_j, \mathcal{M}^j)$ is a well-linked instance;*
- *For all $1 \le j \ne j' \le r$, there is no edge in G with one endpoint in G_j and the other in $G_{j'}$;*
- $\sum_{j=1}^{r} |\mathcal{M}^j| \ge 63|\mathcal{M}|/64$*; and*
- $\left|V(G) \setminus \left(\bigcup_{j=1}^{r} V(G_j)\right)\right| \le \frac{w^*\cdot|\mathcal{M}|}{64}$.

For each $1 \le j \le r$, let $W_j = w^*|\mathcal{M}^j|$ be the contribution of the demand pairs in $\mathcal{M}^j$ to the current flow solution and let $W = \sum_{j=1}^{r} W_j = \Omega(X^*/\log k)$. Let $n_j = |V(G_j)|$, and let N_j be the number of the non-terminal vertices in G_j. The main tool in proving Theorem 4.1 is the following theorem.

Theorem 4.3 *There is an efficient algorithm, that computes, for every $1 \le j \le r$, one of the following:*

1. *Either a collection $\mathcal{P}^j$ of node-disjoint paths, routing $\Omega\left(W_j^{1/19}/\mathrm{poly}\log n\right)$ demand pairs of $\mathcal{M}^j$ in G_j; or*
2. *A collection $\tilde{\mathcal{M}}^j \subseteq \mathcal{M}^j$ of demand pairs, with $|\tilde{\mathcal{M}}^j| \ge |\mathcal{M}^j|/2$, such that $\mathsf{OPT}(G_j, \tilde{\mathcal{M}}^j) \le w^*|\tilde{\mathcal{M}}^j|/8$.*

Before we prove Theorem 4.3, we show that Theorem 4.1 follows from it. We apply Theorem 4.3 to every instance $(G_j, \mathcal{M}^j)$, for $1 \leq j \leq r$. We say that instance $(G_j, \mathcal{M}^j)$ is a type-1 instance, if the first outcome happens for it, and we say that it is a type-2 instance otherwise. Let $I_1 = \left\{j \mid (G_j, \mathcal{M}^j) \text{ is a type-1 instance}\right\}$, and we define I_2 similarly for type-2 instances. We now consider two cases.

The first case happens if $\sum_{j \in I_1} W_j \geq W/2$. We show that in this case, our algorithm returns a routing of $\Omega\left(\frac{X^*}{N^{9/19} \cdot \text{poly} \log n}\right)$ demand pairs. We need the following lemma, whose proof is omitted here. The proof uses standard techniques: namely, we show that the treewidth of each graph G_j is at least $\Omega(W_j/\log k)$, and so G_j must contain a large grid minor.

Lemma 4.1 *For each $1 \leq j \leq r$, $N_j \geq \Omega(W_j^2/\log^2 k)$.*

The number of the demand pairs we route in each type-1 instance $(G_j, \mathcal{M}^j)$ is then at least:

$$\begin{aligned}\Omega\left(\frac{W_j^{1/19}}{\text{poly} \log n}\right) &= \Omega\left(\frac{W_j}{W_j^{18/19} \cdot \text{poly} \log n}\right) \\ &= \Omega\left(\frac{W_j}{\left(\sqrt{N_j} \log k\right)^{18/19} \cdot \text{poly} \log n}\right) \\ &= \Omega\left(\frac{W_j}{N^{9/19} \cdot \text{poly} \log n}\right).\end{aligned}$$

Overall, since $\sum_{j \in I_1} W_j \geq W/2$, the number of the demand pairs routed is $\Omega\left(\frac{W}{N^{9/19} \cdot \text{poly} \log n}\right) = \Omega\left(\frac{X^*}{N^{9/19} \cdot \text{poly} \log n}\right)$.

Consider now the second case, where $\sum_{j \in I_2} W_j \geq W/2$. Let $\mathcal{M}' = \bigcup_{j \in I_2} \tilde{\mathcal{M}}^j$. Then $|\mathcal{M}'| = \sum_{j \in I_2} |\tilde{\mathcal{M}}^j| \geq \sum_{j \in I_2} \frac{|\mathcal{M}^j|}{2} \geq \frac{1}{4} \sum_{j=1}^{r} |\mathcal{M}^j| \geq \frac{|\mathcal{M}|}{8}$. We claim that the following inequality, that is violated by the current LP-solution, is a valid constraint of (LP-flow2):

$$\sum_{(s_i, t_i) \in \mathcal{M}'} x'_i \leq w^* |\mathcal{M}'|/2. \tag{7}$$

In order to do so, it is enough to prove that $\mathsf{OPT}(G, \mathcal{M}') < w^*|\mathcal{M}'|/2$. Assume otherwise, and let $\mathcal{P}^*$ be the optimal solution for instance $(G, \mathcal{M}')$, so $|\mathcal{P}^*| \geq w^*|\mathcal{M}'|/2$. We say that a path $P \in \mathcal{P}^*$ is bad if it contains a vertex of $V(G) \setminus \left(\bigcup_{j=1}^{r} V(G_j)\right)$. The number of such bad paths is bounded by the number of such vertices - namely, at most $\frac{w^* \cdot |\mathcal{M}|}{64} \leq \frac{w^*|\mathcal{M}'|}{8} \leq \frac{|\mathcal{P}^*|}{2}$. Therefore, at least $w^*|\mathcal{M}'|/4$ paths in $\mathcal{P}^*$ are good. Each such path must be contained in one of the graphs G_j corresponding to a type-2 instance. For each $j \in I_2$, let $\hat{\mathcal{M}}^j \subseteq \tilde{\mathcal{M}}^j$ be the set of the demand pairs routed by good paths of $\mathcal{P}^*$. Then, on the one hand, $\sum_{j \in I_2} |\hat{\mathcal{M}}^j| \geq w^*|\mathcal{M}'|/4 = w^* \sum_{j \in I_2} |\tilde{\mathcal{M}}^j|/4$, while, on the other hand, since all demand pairs in $\hat{\mathcal{M}}^j$ can be routed simultaneously in G_j, for all $j \in I_2$, $|\hat{\mathcal{M}}^j| \leq w^*|\tilde{\mathcal{M}}^j|/8$, a contradiction. We conclude that $\mathsf{OPT}(G, \mathcal{M}') < w^*|\mathcal{M}'|/2$, and (7) is a valid constraint of (LP-flow2).

From now on, we focus on proving Theorem 4.3. Since from now on we only consider one instance $(G_j, \mathcal{M}^j)$, for simplicity, we abuse the notation and denote G_j by G, and $\mathcal{M}^j$ by $\mathcal{M}$. As before, we denote $\mathcal{T} = \mathcal{T}(\mathcal{M})$. We denote by $W = w^* \cdot |\mathcal{M}|$ the total amount of flow sent between the demand pairs in the new set $\mathcal{M}$ in the LP solution (note that this is not necessarily a valid LP-solution for the new instance, as some of the flow-paths may use vertices lying outside of G^j). We use n to denote the number of vertices in G. Value k - the number of the demand pairs in the original instance - remains unchanged. Our goal is to either find a collection of node-disjoint paths routing $\Omega(W^{1/19}/\text{poly}(\log(nk)))$ demand pairs of $\mathcal{M}$ in G, or to find a collection $\tilde{\mathcal{M}} \subseteq \mathcal{M}$ of at least $|\mathcal{M}|/2$ demand pairs, such that $\mathsf{OPT}(G, \tilde{\mathcal{M}}) \leq w^*|\tilde{\mathcal{M}}|/8$. We will rely on the fact that all terminals are α_{WL}-well-linked in G, for $\alpha_{\text{WL}} = \Theta(w^*/\log k)$. We assume that G is connected, since otherwise all terminals must be contained in a single connected component of G and we can discard all other connected components.

We assume that we are given an embedding of G into the sphere. For every pair $v, v' \in V(G)$ of vertices, we let $d_{\text{GNC}}(v, v')$ be the length of the shortest G-normal curve connecting v to v' in this embedding, minus 1. It is easy to verify that d_{GNC} is a metric: that is, $d_{\text{GNC}}(v, v) = 0$, $d_{\text{GNC}}(v, v') = d_{\text{GNC}}(v', v)$, and the triangle inequality holds for d_{GNC}. The value $d_{\text{GNC}}(v, v')$ can be computed efficiently, by solving an appropriate shortest path problem instance in the graph dual to G. Given a vertex v and a subset U of vertices of G, we denote by $d_{\text{GNC}}(v, U) = \min_{u \in U} \{d_{\text{GNC}}(v, u)\}$. Similarly, given two subsets U, U' of vertices of G, we denote $d_{\text{GNC}}(U, U') = \min_{u \in U, u' \in U'} \{d_{\text{GNC}}(u, u')\}$. Finally, given a G-normal curve C, and a vertex v in G, we let $d_{\text{GNC}}(v, C) = \min_{u \in V(C)} \{d_{\text{GNC}}(v, u)\}$.

Over the course of the algorithm, we will sometimes select some face of the drawing of G as the outer face and consider the resulting drawing of G in the plane. The function d_{GNC} remains unchanged, and it is only defined with respect to the fixed embedding of G into the sphere.

5. ENCLOSURES, SHELLS, AND TERMINAL SUBSETS

In this section we develop some of the technical machinery that we use in our algorithm, and describe the first steps of the algorithm, starting with enclosures for the terminals.

5.1 Constructing Enclosures

Throughout the algorithm, we use a parameter $\Delta = \left\lceil W^{2/19} \right\rceil$. We assume that $W > \Omega(\Delta)$, since otherwise W is bounded by a constant, and we can return the routing of a single demand pair. The goal of this step is to construct enclosures around the terminals, that are defined below.

Definition 5.1 *An* enclosure *for terminal $t \in \mathcal{T}$ is a simple disc D_t containing the terminal t, whose boundary is denoted by C_t, that has the following properties. (Recall that $V(D_t)$ is the set of all vertices of G contained in D_t.)*

- *C_t is a simple closed G-normal curve with $\ell(C_t) = \Delta$;*
- *$|\mathcal{T} \cap V(D_t)| \leq 4\Delta/\alpha_{\text{WL}}$; and*
- *$V(D_t)$ induces a connected graph in G.*

The following theorem, whose proof is omitted due to lack of space, allows us to construct a collection of enclosures around the terminals with additional useful properties.

Theorem 5.1 *There is an efficient algorithm that constructs an enclosure D_t for every terminal $t \in \mathcal{T}$, such that for all $t, t' \in \mathcal{T}$: (i) if $D_t \subseteq D_{t'}$, then $D_t = D_{t'}$; and (ii) if $D_t \cap D_{t'} = \emptyset$, then there are Δ node-disjoint paths between $V(C_t)$ and $V(C_{t'})$ in G.*

Distances between terminals. For every pair t, t' of terminals, we define the distance $d(t, t')$ between t and t' to be the length of the shortest G-normal open curve, with one endpoint in $V(C_t)$ and another in $V(C_{t'})$. (Notice that if $D_t \cap D_{t'} \neq \emptyset$, then $d(t, t') = 1$).

5.2 Constructing Shells

Suppose we are given some terminal $t \in \mathcal{T}$ and an integer $r \geq 1$. In this section we show how to construct a shell of depth r around t. Shells play a central role in our algorithm. In order to construct the shell, we need to fix a plane drawing of the graph G, by choosing one of the faces F_t of the drawing of G on the sphere as the outer face. The choice of the face F_t will affect the construction of the shell, but once the face F_t is fixed, the shell construction is fixed as well. We require that for every vertex v on the boundary of F_t, $d_{\text{GNC}}(v, C_t) \geq r + 1$, and that C_t separates all vertices on the boundary of F_t from t. We note that when we construct shells for different terminals t, t', we may choose different faces $F_t, F_{t'}$, and thus obtain different embeddings of G into the plane. We now define a shell.

Definition 5.2 *Suppose we are given a terminal $t \in \mathcal{T}$, a face F_t in the drawing of G on the sphere, and an integer r, such that for every vertex v on the boundary of F_t, $d_{\text{GNC}}(v, C_t) \geq r + 1$, and C_t separates t from the boundary of F_t.*
A shell $\mathcal{Z}^r(t)$ of depth r around t with respect to F_t is a collection $\mathcal{Z}^r(t) = (Z_1(t), Z_2(t), \ldots, Z_r(t))$ of r tight concentric cycles around C_t. In other words, all cycles $Z_h(t)$ are simple and disjoint from each other, and the following properties hold. For each $1 \leq h \leq r$, let $D(Z_h(t))$ be the disc whose boundary is $Z_h(t)$ in the planar drawing of G with F_t as the outer face. Then:

- *J1. $D_t \subsetneq D(Z_1(t)) \subsetneq D(Z_2(t)) \subsetneq \cdots \subsetneq D(Z_r(t))$; and*
- *J2. for every $1 \leq h \leq r$, if H is the graph obtained from G by contracting all vertices lying in $D(Z_{h-1}(t))$ into a super-node a, then $Z_h(t) = \text{min-cycle}(H, a)$ (when $h = 1$, we contract D_t into a super-node a).*

Notice that from this definition we immediately obtain the following additional properties:

- J3. For every $1 \leq h \leq r$, for every vertex $v \in V(Z_h(t))$, there is a G-normal curve of length 2 connecting v to some vertex of $V(Z_{h-1}(t))$ (or to a vertex of $V(C_t)$ if $h = 1$).
- J4. For every $1 \leq h \leq r$, for every vertex $v \in V(Z_h(t))$, there is a G-normal curve $\gamma(v)$ of length $h + 1$ connecting v to some vertex of $V(C_t)$, so that $\gamma(v) \subseteq D(Z_h(t))$.

We also need the following two observations.

Observation 5.1 *For all $1 \leq h \leq r$, if $\tilde{U}_h$ is the set of vertices of G lying in $D(Z_h(t))$, then $G[\tilde{U}_h]$ is connected.*

Observation 5.2 *Let t, t' be any pair of terminals, and let $r, r' > 0$ be any integers, such that $d(t, t') > r + r' + 1$. Let $\mathcal{Z}^r(t) = (Z_1(t), \ldots, Z_r(t))$ be a shell of depth r around t with respect to some face F_t, and let $\mathcal{Z}^{r'}(t') = (Z_1(t') \ldots, Z_{r'}(t'))$ be a shell of depth r' around t' with respect to some face $F_{t'}$. Then for all $1 \leq h \leq r$ and $1 \leq h' \leq r'$, $Z_h(t) \cap Z_{h'}(t') = \emptyset$.*

5.3 Terminal Subsets

Let $\Delta_0 = 20(\left\lceil \log_{10/9} n \right\rceil + 1)\Delta = \Theta(\Delta \log n)$. Our next step is to define a family of disjoint subsets of terminals, so that the terminals within each subset are close to each other, while the terminals belonging to different subsets are far enough from each other. We will ensure that almost all terminals of $\mathcal{T}$ belong to one of the resulting subsets. The proof of the following theorem uses standard techniques and is omitted from this extended abstract.

Theorem 5.2 *There is an efficient algorithm to compute a collection $\mathcal{X} = \{X_1, \ldots, X_q\}$ of disjoint subsets of terminals of $\mathcal{T}$, such that:*

- *for each $1 \leq i \leq q$, for every pair $t, t' \in X_i$ of terminals, $d(t, t') \leq \Delta_0$;*
- *for all $1 \leq i \neq j \leq q$, for every pair $t \in X_i$, $t' \in X_j$ of terminals $d(t, t') \geq 5\Delta$; and*
- *$\sum_{i=1}^{q} |X_i| \geq 0.9|\mathcal{T}|$.*

We use a parameter $\tau = W^{18/19}$. We say that a set $X \in \mathcal{X}$ of terminals is *heavy* if $w^*|X| \geq \tau$, and we say that it is *light* otherwise. We say that a demand pair (s, t) is heavy iff both s and t belong to heavy subsets of terminals in $\mathcal{X}$. We say that it is light if at least one of the two terminals belongs to a light subset, and the other belongs to some set of $\mathcal{X}$. Note that a demand pair (s, t) may be neither heavy nor light, for example, if s or t lie in $\mathcal{T} \setminus \bigcup_{X \in \mathcal{X}} X$. Let $\mathcal{M}_0$ be the set of all demand pairs that are neither heavy nor light. Then $|\mathcal{M}_0| \leq 0.2|\mathcal{M}|$. We say that Case 1 happens if there are at least $0.1|\mathcal{M}|$ light demand pairs, and we say that Case 2 happens otherwise. Notice that in Case 2, at least $0.7|\mathcal{M}|$ of the demand pairs are heavy. In the next two sections we handle Case 1 and Case 2 respectively.

6. CASE 1: LIGHT DEMAND PAIRS

Let $\mathcal{M}^L \subseteq \mathcal{M}$ be the set of all light demand pairs, so $|\mathcal{M}^L| \geq 0.1|\mathcal{M}|$. We assume w.l.o.g. that for every pair $(s, t) \in \mathcal{M}^L$, t belongs to a light set in $\mathcal{X}$. Let $S^L, T^L \subseteq \mathcal{T}$ be the sets of the source and the destination vertices of the demand pairs in $\mathcal{M}^L$, respectively, and let $\mathcal{L} \subseteq \mathcal{X}$ be the set of all light terminal subsets. Recall that we have assumed that every terminal participates in exactly one demand pair. The goal of this section is to prove the following theorem.

Theorem 6.1 *Let $p^* = \Theta\left(\frac{\alpha_{\text{WL}}|\mathcal{M}^L|}{\tau \log n}\right)$. There is an efficient algorithm, that computes a set of node-disjoint paths in G, routing at least $\min\left\{\Omega(p^*), \Omega\left(\frac{\Delta}{p^* \log n}\right)\right\}$ demand pairs.*

We first show that Theorem 6.1 concludes the proof of Theorem 4.3 for Case 1. Indeed, since $|\mathcal{M}^L| \geq 0.1|\mathcal{M}|$, we get that $p^* = \Theta\left(\frac{\alpha_{\text{WL}}|\mathcal{M}|}{\tau \log n}\right) = \Theta\left(\frac{w^*|\mathcal{M}|}{W^{18/19}\log n \log k}\right) = \Theta\left(\frac{W^{1/19}}{\log n \log k}\right)$. Notice that $\Omega\left(\frac{\Delta}{p^* \log n}\right) = \Omega\left(\frac{W^{2/19}\log k}{W^{1/19}}\right) = \Omega\left(W^{1/19}\log k\right)$. Therefore, the algorithm routes $\Omega\left(\frac{W^{1/19}}{\log n \log k}\right)$ demand pairs via node-disjoint paths. The rest of this section is devoted to proving Theorem 6.1.

Our first step is to compute a large subset $\mathcal{M}_0 \subseteq \mathcal{M}^L$ of light demand pairs, so that, if we denote by S_0 and T_0 the sets of the source and the destination vertices of the demand pairs in $\mathcal{M}_0$, then there is a set $\mathcal{Q}$ of $|\mathcal{M}_0|$ node-disjoint paths connecting the vertices of S_0 to a subset of vertices of T^L, that we denote by T'. Additionally, we ensure that every terminal set $X \in \mathcal{L}$, $|X \cap T'| \leq 1$, and $|X \cap T_0| \leq 1$. We note that the sets S_0 and T' do not necessarily form demand pairs. We will eventually route a subset of the pairs of $\mathcal{M}_0$.

Theorem 6.2 *There is an efficient algorithm to compute a subset $\mathcal{M}_0 \subseteq \mathcal{M}^L$ of $\kappa_0 = \Theta\left(\frac{\alpha_{\text{WL}}|\mathcal{M}^L|}{\tau}\right)$ demand pairs, and a subset $T' \subseteq T^L$ of κ_0 terminals, such that, if we denote by S_0 and T_0 the sets of the source and the destination vertices of the demand pairs in $\mathcal{M}_0$, then:*

- *There is a set $\mathcal{Q}$ of κ_0 node-disjoint paths connecting the vertices of S_0 to the vertices of T'; and*
- *For each set $X \in \mathcal{L}$, $|X \cap T'| \leq 1$, and $|X \cap T_0| \leq 1$.*

We assume that $|\mathcal{M}_0| \geq 1000$, as otherwise we can route a single demand pair, and obtain a solution routing $\Omega\left(\frac{\alpha_{\text{WL}}|\mathcal{M}^L|}{\tau}\right)$ demand pairs.

Recall that every set $X \in \mathcal{L}$ contains at most one terminal from T_0. Since $|S_0| = |T_0|$, there is some set $X_0 \in T_0$, that contains exactly one terminal $t_0 \in T_0$, and at most one additional terminal from S_0. We will view t_0 as our "center" terminal, and we discard from $\mathcal{M}_0$ the demand pairs in which t_0 and the terminal in $S_0 \cap X_0$ (if it exists) participate. The main tool in our algorithm for Case 1 is a crossbar, that we define below. Let $\Delta_1 = \lfloor \Delta/6 \rfloor$ and $\Delta_2 = \lfloor \Delta_1/3 \rfloor$. Given a shell $\mathcal{Z}(t) = (Z_1(t), \ldots, Z_{\Delta_1}(t))$ of depth Δ_1 around some terminal t, we will always denote by $D^*(t) = D(Z_{\Delta_1}(t))$, and by $\tilde{D}(t) = D(Z_{\Delta_2}(t))$. We will view the cycles $Z_1(t), \ldots, Z_{\Delta_2}(t)$ as the "inner" part of the shell $\mathcal{Z}(t)$. The crossbar is defined with respect to some subset $\mathcal{M}^* \subseteq \mathcal{M}_0$ of demand pairs (see Figure 3).

Definition 6.1 *Suppose we are given a subset $\mathcal{M}^* \subseteq \mathcal{M}_0$ of demand pairs and an integer $p \geq 1$. Let S^* and T^* be the sets of all source and all destination vertices participating in the demand pairs of $\mathcal{M}^*$, respectively. A p-crossbar for $\mathcal{M}^*$ consists of:*

- *For each terminal $t \in T^* \cup \{t_0\}$, a shell $\mathcal{Z}(t)$ of depth Δ_1 around t, such that for all $t, t' \in T^* \cup \{t_0\}$, $D^*(t) \cap D^*(t') = \emptyset$, and for all $s \in S^*$ and $t' \in T^* \cup \{t_0\}$, $s \notin D^*(t')$; and*
- *For each $v \in S^* \cup T^*$, a collection $\mathcal{P}(v)$ of paths, where:*
  - *For each $s \in S^*$, $\mathcal{P}(s)$ contains exactly one path, connecting s to a vertex of C_{t_0};*
  - *For each $t \in T^*$, $\mathcal{P}(t)$ contains exactly p paths, where each path connects a vertex of C_t to a vertex of C_{t_0}; and*
  - *All paths in $\mathcal{P} = \bigcup_{v \in S^* \cup T^*} \mathcal{P}(v)$ are node-disjoint from each other.*

In order to route a large subset of the demand pairs in $\mathcal{M}^*$, we need a crossbar with slightly stronger properties, that we call a *good crossbar*, and define below. We say that a path P is monotone with respect to a cycle C iff $P \cap C$ is a path.

Definition 6.2 *Given a set $\mathcal{M}^* \subseteq \mathcal{M}_0$ of demand pairs, where S^*, T^* are the sets of the source and the destination vertices of the demand pairs in $\mathcal{M}^*$ respectively, and an integer $p > 1$, a p-crossbar $\left(\{\mathcal{Z}(t)\}_{t \in T^* \cup \{t_0\}}, \{\mathcal{P}(v)\}_{v \in S^* \cup T^*}\right)$ is a* good crossbar, *if the following additional properties hold:*

C1. For all $t \in T^*$ and all $v \in (S^* \cup T^*) \setminus \{t\}$, all paths in $\mathcal{P}(v)$ are disjoint from $\tilde{D}(t)$.

C2. For all $t \in T^*$, all paths in $\mathcal{P}(t)$ are monotone with respect to $(Z_1(t), \ldots, Z_{\Delta_2}(t))$. Also, for all $v \in S^* \cup T^*$, all paths in $\mathcal{P}(v)$ are monotone with respect to $(Z_1(t_0), \ldots, Z_{\Delta_2}(t_0))$.

C3. There is a partition $\Sigma = \{\sigma(v) \mid v \in S^* \cup T^*\}$ of $Z_{\Delta_2}(t_0)$ into $|S^*| + |T^*|$ disjoint segments, such that for all $v, v' \in S^* \cup T^*$ with $v \neq v'$, $\sigma(v) \cap \mathcal{P}(v') = \emptyset$.

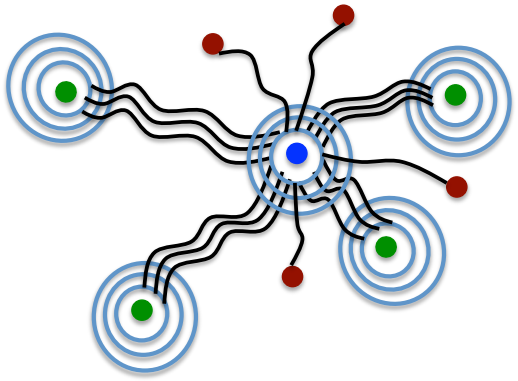

Figure 3: A crossbar. The center vertex t_0 is shown in blue, the vertices of S^* in red, and the vertices of T^* in green.

The following theorem shows that, given a p-crossbar in G, where p is large enough, we can route many demand pairs in $\mathcal{M}^*$. We omit the proof due to lack of space.

Theorem 6.3 *Suppose we are given a subset $\mathcal{M}^* \subseteq \mathcal{M}_0$ of κ demand pairs, where S^* and T^* are the sets of all source and all destination vertices of the demand pairs in $\mathcal{M}^*$, respectively. Assume further that we are given a good p-crossbar $\left(\{\mathcal{Z}(t)\}_{t \in T^* \cup \{t_0\}}, \{\mathcal{P}(v)\}_{v \in S^* \cup T^*}\right)$ for $\mathcal{M}^*$, and let $q = \min\{\Delta_2, \lfloor (p-1)/2 \rfloor, \lceil \kappa/2 \rceil\}$. Then there is an efficient algorithm that routes at least q demand pairs in $\mathcal{M}^*$ via node-disjoint paths in G.*

In order to complete the proof of Theorem 6.1, we use the following theorem, whose proof is omitted from this extended abstract, that allows us to construct a good crossbar.

Theorem 6.4 *There is an efficient algorithm that either finds a routing of $\Omega(\kappa_0)$ demand pairs in $\mathcal{M}_0$, or computes a subset $\mathcal{M}^* \subseteq \mathcal{M}_0$ of $\Omega(\kappa_0/\log n)$ demand pairs, and a good p-crossbar for $\mathcal{M}^*$, with $p = \Omega(\Delta/\kappa_0)$.*

Recall that $\kappa_0 = \Theta\left(\frac{\alpha_{\text{WL}}|\mathcal{M}^L|}{\tau}\right)$. Letting $p^* = \frac{\kappa_0}{\log n} = \Omega\left(\frac{\alpha_{\text{WL}}|\mathcal{M}^L|}{\tau \log n}\right)$, from Theorem 6.3, we can efficiently find a routing of at least $\min\left\{\Omega(p^*), \Omega\left(\frac{\Delta}{p^* \log n}\right)\right\}$ demand pairs, concluding the proof of Theorem 6.1.

7. CASE 2: HEAVY DEMAND PAIRS.

In this case, we assume that at least $0.7|\mathcal{M}|$ demand pairs are heavy. Let $\mathcal{H} = \{X_1, \ldots, X_q\} \subseteq \mathcal{X}$ be the collection of all heavy subsets of terminals, so $q \leq 2W/\tau$, and let $\mathcal{M}^h$ be the set of all heavy demand pairs, so for all $(s,t) \in \mathcal{M}^h$, both s and t lie in the sets of $\mathcal{H}$. We partition the set $\mathcal{M}^h$ of demand pairs into q^2 subsets, where for $1 \leq i,j \leq q$, set $\mathcal{M}_{i,j}$ contains all demand pairs (s,t) with $s \in X_i$ and $t \in X_j$ (notice that it is possible that $i = j$). We then find an approximate solution to each resulting problem separately. The main theorem of this section is the following.

Theorem 7.1 *There is an efficient algorithm, that for each $1 \leq i,j \leq q$, computes a subset $\mathcal{M}'_{i,j} \subseteq \mathcal{M}_{i,j}$ of at least $5|\mathcal{M}_{i,j}|/6$ demand pairs, and a collection $\mathcal{P}_{i,j}$ of node-disjoint paths routing a subset of the demand pairs in $\mathcal{M}'_{i,j}$ in G, with $|\mathcal{P}_{i,j}| \geq \min\left\{\frac{\mathsf{OPT}(G,\mathcal{M}'_{i,j})}{c_1\Delta_0^8\log^3 n}, \frac{\alpha_{\text{WL}}\cdot|\mathcal{M}'_{i,j}|}{c_2\Delta_0^2}\right\}$, for some universal constants c_1 and c_2.*

Before we prove this theorem, we show that it concludes the proof of Theorem 4.3 for Case 2. Let set π contain all pairs (i,j) with $1 \leq i,j \leq q$, such that $|\mathcal{M}_{i,j}| \geq 0.1|\mathcal{M}|/q^2$, and let $\tilde{\mathcal{M}} = \bigcup_{(i,j)\in\pi}\mathcal{M}_{i,j}$. Since the total number of heavy demand pairs is at least $0.7|\mathcal{M}|$, it is easy to verify that $|\tilde{\mathcal{M}}| \geq 0.6|\mathcal{M}|$.

We apply Theorem 7.1, to compute, for each $(i,j) \in \pi$, the subset $\mathcal{M}'_{i,j} \subseteq \mathcal{M}_{i,j}$ of at least $5|\mathcal{M}_{i,j}|/6$ demand pairs and the corresponding set $\mathcal{P}_{i,j}$ of paths routing a subset of the demand pairs in $\mathcal{M}'_{i,j}$. Let $\tilde{\mathcal{M}}' \subseteq \tilde{\mathcal{M}}$ be the set of all demand pairs in $\bigcup_{(i,j)\in\pi}\mathcal{M}'_{i,j}$. Then:

$$|\tilde{\mathcal{M}}'| = \sum_{(i,j)\in\pi}|\mathcal{M}'_{i,j}| \geq \sum_{(i,j)\in\pi} 5|\mathcal{M}_{i,j}|/6 = 5|\tilde{\mathcal{M}}|/6 \geq |\mathcal{M}|/2.$$

If, for any $(i,j) \in \pi$, we obtain a solution with $|\mathcal{P}_{i,j}| \geq \frac{W}{2^{13}\cdot\alpha_{\text{AKR}}\cdot c_1c_2q^2\Delta_0^8\log^3 n\cdot\log k}$, then we return the set $\mathcal{P}_{i,j}$ as our final solution. Substituting $\Delta_0 = O(\Delta\log n)$, $\Delta = \left\lceil W^{2/19}\right\rceil$, $q = O(W/\tau)$, and $\tau = W^{18/19}$, we get that:

$$|\mathcal{P}_{i,j}| \geq \Omega\left(\frac{W\tau^2}{W^2\Delta^8\log^{11} n\log k}\right) \geq \Omega\left(\frac{W^{36/19}}{W\cdot W^{16/19}\log^{11} n\log k}\right) = \Omega\left(\frac{W^{1/19}}{\log^{11} n\log k}\right).$$

Otherwise, for all $(i,j) \in \pi$, the resulting solution $|\mathcal{P}_{i,j}| < \frac{W}{2^{13}\cdot\alpha_{\text{AKR}}\cdot c_1c_2q^2\Delta_0^8\log^3 n\cdot\log k}$. We then return the subset $\tilde{\mathcal{M}}'$ of demand pairs. As observed above, $|\tilde{\mathcal{M}}'| \geq |\mathcal{M}|/2$, so it is now enough to show that $\mathsf{OPT}(G,\tilde{\mathcal{M}}') \leq w^*|\tilde{\mathcal{M}}'|/8$.

Assume otherwise, and let $\mathcal{P}^*$ be a solution to instance $(G,\tilde{\mathcal{M}}')$, routing a subset $\mathcal{M}^* \subseteq \tilde{\mathcal{M}}'$ of at least $w^*|\tilde{\mathcal{M}}'|/8 \geq w^*|\mathcal{M}|/16$ demand pairs. Then there is a pair of indices $(i,j) \in \pi$, such that $|\mathcal{M}^* \cap \mathcal{M}'_{i,j}| \geq \frac{w^*|\mathcal{M}|}{16q^2}$. Therefore, $\mathsf{OPT}(G,\mathcal{M}'_{i,j}) \geq \frac{w^*|\mathcal{M}|}{16q^2}$. From Theorem 7.1, we compute a set $\mathcal{P}_{i,j}$ of paths, routing a subset of demand pairs of $\mathcal{M}'_{i,j}$, with either $|\mathcal{P}_{i,j}| \geq \frac{\mathsf{OPT}(G,\mathcal{M}'_{i,j})}{c_1\Delta_0^8\log^3 n}$, or $|\mathcal{P}_{i,j}| \geq \frac{\alpha_{\text{WL}}\cdot|\mathcal{M}'_{i,j}|}{c_2\Delta_0^2}$. In the former case,

$$|\mathcal{P}_{i,j}| \geq \frac{\mathsf{OPT}(G,\mathcal{M}'_{i,j})}{c_1\Delta_0^8\log^3 n} \geq \frac{w^*|\mathcal{M}|}{16c_1q^2\Delta_0^8\log^3 n} = \frac{W}{16c_1q^2\Delta_0^8\log^3 n},$$

while in the latter case, observe that $|\mathcal{M}'_{i,j}| \geq 5|\mathcal{M}_{i,j}|/6 \geq |\mathcal{M}|/(12q^2)$ from the definition of π and $\tilde{M}$. Therefore,

$$|\mathcal{P}_{i,j}| \geq \frac{\alpha_{\text{WL}}\cdot|\mathcal{M}'_{i,j}|}{c_2\Delta_0^2} \geq \frac{w^*\cdot|\mathcal{M}|}{12\cdot 512\cdot\alpha_{\text{AKR}}\cdot c_2q^2\Delta_0^2\log k} > \frac{W}{2^{13}\cdot\alpha_{\text{AKR}}\cdot c_1c_2q^2\Delta_0^8\log^2 n\log k},$$

a contradiction.

From now on we focus on proving Theorem 7.1. We fix a pair of indices $1 \leq i,j \leq q$. In order to simplify the notation, we denote $\mathcal{M}_{i,j}$ by $\mathcal{N}$, X_i by X and X_j by Y. Our goal is to compute a subset $\mathcal{N}' \subseteq \mathcal{N}$ of at least $5|\mathcal{N}|/6$ demand pairs, together with a set $\mathcal{P}$ containing at least $\min\left\{\Omega\left(\frac{\mathsf{OPT}(G,\mathcal{N}')}{\Delta_0^8\log^3 n}\right), \Omega\left(\frac{\alpha_{\text{WL}}|\mathcal{N}'|}{\Delta_0^2}\right)\right\}$ paths, routing a subset of the demand pairs in $\mathcal{N}'$. Let $x \in X$ and $y \in Y$ be any pair of terminals. Recall that for every terminal $t \in \mathcal{T}(\mathcal{N}) \cap X$, $d(t,x) \leq \Delta_0$, and for every terminal $t \in \mathcal{T}(\mathcal{N}) \cap Y$, $d(t,y) \leq \Delta_0$. We consider two subcases. The first subcase happens when $d(x,y) > 5\Delta_0$, and otherwise the second subcase happens, so the second subcase includes the case where $X = Y$.

7.1 Subcase 2a: $d(x,y) > 5\Delta_0$

In this case, we set $\mathcal{N}' = \mathcal{N}$. We will compute a set $\mathcal{P}$ of at least $\Omega\left(\frac{\mathsf{OPT}(G,\mathcal{N})}{\Delta_0^6\log n}\right)$ node-disjoint paths, routing a subset of the demand pairs in $\mathcal{N}$. We start by defining a simpler special case of the problem, and show that we can find a good approximation algorithm for this special case. The special case is somewhat similar to routing on a cylinder, and we solve it by reducing it to this setting.

7.1.1 A Special Case

Suppose we are given a connected planar graph $\hat{G}$ embedded on the sphere, and two disjoint simple cycles Z, Z' in $\hat{G}$. Suppose also that we are given a set $\hat{\mathcal{M}}$ of demand pairs, where all source vertices lie on Z and all destination vertices lie on Z' (we note that the same vertex may participate in a number of demand pairs). Let $D(Z), D(Z')$ be two discs with boundaries Z and Z', respectively, so that $D(Z) \cap D(Z') = \emptyset$. Assume additionally that we are given a closed $\hat{G}$-normal curve C of length at most Δ, that is contained in $D^\circ(Z)$, so that for every vertex $v \in Z$, there is a $\hat{G}$-normal curve $\gamma(v)$ of length at most $2\Delta_0$ connecting v to a vertex of C, and $\gamma(v)$ is internally disjoint from Z and

C. Similarly, assume that we are given a closed $\hat{G}$-normal curve C' of length at most Δ, that is contained in $D^{\circ}(Z')$, so that for every vertex $v' \in Z'$ there is a $\hat{G}$-normal curve $\gamma(v')$ of length at most $2\Delta_0$, connecting v' to a vertex of C', and $\gamma(v')$ is internally disjoint from Z' and C' (see Figure 4). This finishes the definition of the special case. The following theorem shows that we can obtain a good approximation for it. The proof is omitted from this extended abstract.

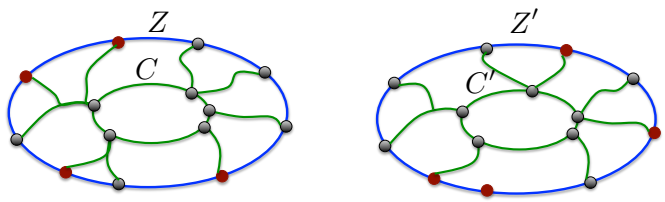

Figure 4: The special case, with the terminals shown in red

Theorem 7.2 *There is an efficient algorithm, that, given any instance $(\hat{G}, \hat{\mathcal{M}})$ of the* NDP *problem as above, computes a solution of value at least $\Omega\left(\frac{OPT(\hat{G},\hat{\mathcal{M}})}{\Delta_0^2 \log n}\right)$.*

7.1.2 Completing the Proof

We now complete the proof of Theorem 7.1 for Case 2a, by reducing it to the special case defined above. We assume that $\mathsf{OPT}(G,\mathcal{N}) > 2^{13}\Delta_0^4$, since otherwise we can route a single demand pair and obtain a valid solution. We denote by S and T the sets of all source and all destination vertices of the demand pairs in $\mathcal{N}$, respectively.

Our first step is to construct shells $\mathcal{Z}(x) = (Z_1(x), \ldots, Z_{2\Delta_0}(x))$ and $\mathcal{Z}(y) = (Z_1(y), \ldots, Z_{2\Delta_0}(y))$ of depth $2\Delta_0$ around x and y, respectively. We would like to ensure that disc $D(Z_{2\Delta_0}(x))$ contains all terminals of X and no terminals of Y, and similarly, disc $D(Z_{2\Delta_0}(y))$ contains all terminals of Y and no terminals of X. In order to ensure this, when constructing the shell $\mathcal{Z}(x)$, we let the face F_x (that is viewed as the outer face in the plane embedding of G when constructing the shell) be the face incident on the terminal y, and similarly we let F_y be the face incident on x (recall that $d(x,y) > 5\Delta_0$, so this choice of faces is consistent with the requirement that the shell depth is bounded by $\min\{d_{\text{GNC}}(v, C_x)\} - 1$ over all vertices v lying on the boundary of F_x). Let $\mathcal{Z}(x) = (Z_1(x), \ldots, Z_{2\Delta_0}(x))$ and $\mathcal{Z}(y) = (Z_1(y), \ldots, Z_{2\Delta_0}(y))$ be the resulting shells.

Claim 7.1 *Disc $D(Z_{2\Delta_0}(x))$ contains all terminals of X and no terminals of Y, and similarly, disc $D(Z_{2\Delta_0}(y))$ contains all terminals of Y and no terminals of X. Moreover, $D(Z_{2\Delta_0}(x)) \cap D(Z_{2\Delta_0}(y)) = \emptyset$.*

For consistency of notation, we will denote $Z_0(x) = C_x$ and $Z_0(y) = C_y$, even though both C_x and C_y are G-normal curves and not cycles.

Let $U = \bigcup_{h=0}^{2\Delta_0} V(Z_h(x))$, and let $U' = \bigcup_{h'=0}^{2\Delta_0} V(Z_{h'}(y))$. For each $1 \le h \le 2\Delta_0$, let U_h be the set of vertices lying in $D^{\circ}(Z_h(x)) \setminus D(Z_{h-1}(x))$, and let $\mathcal{R}_h$ be the set of all connected components of $G[U_h]$. For each $1 \le h' \le 2\Delta_0$, we define $U'_{h'}$ and $\mathcal{R}'_{h'}$ with respect to the shell $\mathcal{Z}(y)$ similarly. Let $\mathcal{R} = \bigcup_{h=1}^{2\Delta_0} \mathcal{R}_h$ and let $\mathcal{R}' = \bigcup_{h'=1}^{2\Delta_0} \mathcal{R}'_{h'}$.

Our next step is to define a mapping $\beta : S \to 2^U$ of all source vertices in S to subsets of vertices of U, and a mapping $\beta' : T \to 2^{U'}$ of all destination vertices in T to subsets of vertices of U'. Every vertex in $S \cup T$ will be mapped to a subset of at most three vertices. We will then replace each demand pair (s,t) with the set $\beta(s) \times \beta'(t)$ of demand pairs. Eventually, for every pair $0 \le h, h' \le 2\Delta_0$ of indices, we will define a subset $\tilde{\mathcal{M}}_{h,h'}$ of the new demand pairs, containing all pairs whose sources lie on $Z_h(x)$ and destinations lie on $Z_{h'}(y)$, to obtain an instance of the special case, that will then be solved using Theorem 7.2.

We start by defining the mapping of the sources. First, for every source vertex $s \in S \cap U$, we set $\beta(s) = \{s\}$. Next, fix any vertex $v^* \in V(C_x)$. For every source vertex $s \in V(D_x) \setminus V(C_x)$, we set $\beta(s) = \{v^*\}$. Finally, consider some component $R \in \mathcal{R}_h$, for some $1 \le h \le 2\Delta_0$, and let $s \in S \cap V(R)$ be any source lying in R. If R has at most three neighbors in $G \setminus R$, then we let $\beta(s)$ be the set of these three neighbors. Otherwise, R has at most one neighbor in $Z_h(x)$, and at least three neighbor in $Z_{h-1}(x)$. We then let $\beta(s)$ contain a single vertex, which is a neighbor of R lying on $Z_{h-1}(x)$. We define the mapping $\beta' : T \to 2^{U'}$ for the destination vertices $t \in T$ similarly.

Let $\tilde{\mathcal{M}} = \bigcup_{(s,t)\in\mathcal{N}} \beta(s) \times \beta'(t)$. In the following two theorems, we show that the problems $(G,\mathcal{N})$ and $(G,\tilde{\mathcal{M}})$ are equivalent to within relatively small factors. The proofs are omitted due to lack of space.

Theorem 7.3 *There is an efficient algorithm, that, given any solution to instance $(G,\tilde{\mathcal{M}})$, that routes κ demand pairs, finds a solution to instance $(G,\mathcal{N})$, routing at least $\frac{\kappa}{21\Delta_0}$ demand pairs.*

Theorem 7.4 $\mathsf{OPT}(G,\tilde{\mathcal{M}}) \ge \frac{\mathsf{OPT}(G,\mathcal{N})}{21\Delta_0}$.

For each $0 \le h, h' \le 2\Delta_0$, let $\tilde{\mathcal{M}}_{h,h'} \subseteq \tilde{\mathcal{M}}$ be the set of all demand pairs $(\tilde{s},\tilde{t})$ with $\tilde{s} \in Z_h(x)$ and $\tilde{t} \in Z_{h'}(y)$.

If $h = 0$ or $h' = 0$, then, since $|V(C_x)|, |V(C_y)| \le \Delta$, $\mathsf{OPT}(G,\tilde{\mathcal{M}}_{h,h'}) \le \Delta$. We route any demand pair in $\tilde{\mathcal{M}}_{h,h'}$ to obtain a factor-Δ approximation to the problem $(G,\tilde{\mathcal{M}}_{h,h'})$.

If both $h, h' > 0$, then we apply Theorem 7.2 to obtain a collection $\mathcal{P}_{h,h'}$ of at least $\Omega\left(\frac{\mathsf{OPT}(G,\tilde{\mathcal{M}}_{h,h'})}{\Delta_0^2 \log n}\right)$ disjoint paths, routing demand pairs in $\tilde{\mathcal{M}}_{h,h'}$. We then take the best among all resulting solutions.

Notice that $\left\{\tilde{\mathcal{M}}_{h,h'} \mid 0 \le h, h' \le 2\Delta_0\right\}$ partition the set $\tilde{\mathcal{M}}$ of demand pairs, and so there is a pair $0 \le h, h' \le 2\Delta_0$ of indices with $\mathsf{OPT}(G,\tilde{\mathcal{M}}_{h,h'}) \ge \frac{\mathsf{OPT}(G,\tilde{\mathcal{M}})}{(2\Delta_0+1)^2} \ge \Omega\left(\frac{\mathsf{OPT}(G,\mathcal{N})}{\Delta_0^3}\right)$.

Therefore, we obtain a routing of at least $\Omega\left(\frac{\mathsf{OPT}(G,\mathcal{N})}{\Delta_0^6 \log n}\right)$ demand pairs.

7.2 Subcase 2b: $d(x,y) \le 5\Delta_0$

Due to lack of space, we only provide an informal overview of the proof of Theorem 7.1 for Case 2b. We start again by defining a special case of the problem, which is similar to the problem of routing on a disc. We show an approximation algorithm for this special case that reduces it to the problem of routing on a disc, and we later use this special case in order to handle the general problem in Case 2b. The remainder of the algorithm follows the general outline of the algorithm for

Case 2a: we construct a shell around the terminal x, map all terminals to the vertices participating in the cycles of the shell, and then reduce the resulting instance to the special case. However, Case 2b is technically more challenging than Case 2a, for several reasons. First, when defining the shell $\mathcal{Z}(x)$ around x, it is now possible that some connected component $R \in \mathcal{R}$ may contain many demand pairs. In this case, our mapping of the terminals to the vertices of the shell is no longer guaranteed to approximately preserve the optimal solution. Therefore, the construction of the shell is more involved technically, and we ensure that each resulting component $R \in \mathcal{R}$ contains relatively few terminals. We then attempt to route demand pairs contained in the components of $\mathcal{R}$ – at most one demand pair per component. If we manage to route a large enough number of demand pairs, then we terminate the algorithm. Otherwise, there are relatively few demand pairs that are contained in the components of $\mathcal{R}$, and we will ignore them for the remainder of the algorithm. The step that maps the remaining terminals to the vertices lying on the cycles of the shells is very similar to the one in Case 2a. The reduction to the special case is more challenging, since we now need to ensure that all terminals lie on a single cycle of the shell. This is done by first selecting two cycles $Z_i(x), Z_j(x)$ of the shell, so that a large number of demand pairs have a source lying on $Z_i(x)$ and a destination lying on $Z_j(x)$, and then carefully moving the terminals lying on $Z_i(x)$ to $Z_j(x)$.

8. PROOF OF THEOREM 1.3

We perform a transformation to instance $(G, \mathcal{M})$ as before, to ensure that every terminal participates in at most one demand pair, and the degree of every terminal is 1. The number of vertices in the new instance is bounded by $2n^2$, and abusing the notation we denote this number by n. We use the following analogue of Theorem 4.1.

Theorem 8.1 *There is an efficient algorithm, that, given any semi-feasible solution to (LP-flow2), either computes a routing of at least $\Omega\left(\frac{(X^*)^{1/19}}{\text{poly}\log n}\right)$ demand pairs in $\mathcal{M}$ via node-disjoint paths, or returns a constraint of type (5), that is violated by the current solution.*

Using the same reasoning as in the proof of Theorem 1.2, it is easy to verify that the above theorem implies Theorem 1.3. We now focus on proving Theorem 8.1.

We again process the fractional solution (x, f) to obtain a new fractional solution (x', f'), where every demand pair sends either 0 or w^* flow units, in the same way as described in Section 4. We let $\mathcal{M}' \subseteq \mathcal{M}$ denote the set of the demand pairs (s_i, t_i) with non-zero flow value x'_i in this new solution. As before, the total flow between the demand pairs in $\mathcal{M}'$ is at least $\Omega(X^*/\log k)$ in the new solution, and, if we find a subset $\mathcal{M}'' \subseteq \mathcal{M}'$ of demand pairs with $\mathsf{OPT}(G, \mathcal{M}'') \leq w^*|\mathcal{M}''|/2$, then set $\mathcal{M}''$ defines a violated constraint of type (5) for (LP-flow2). Therefore, we focus on set $\mathcal{M}'$ and for simplicity denote $\mathcal{M} = \mathcal{M}'$.

We decompose the input instance $(G, \mathcal{M})$ into a collection of well-linked instances $\left\{(G_j, \mathcal{M}^j)\right\}_{j=1}^{r}$ using Theorem 4.2. For each $1 \leq j \leq r$, let $W_j = w^*|\mathcal{M}^j|$ be the contribution of the demand pairs in $\mathcal{M}^j$ to the current flow solution and let $W = \sum_{j=1}^{r} W_j = \Omega(X^*/\log k)$. Theorem 4.3 guarantees that for each $1 \leq j \leq r$, we can obtain one of the following: either (i) a collection $\mathcal{P}^j$ of node-disjoint paths, routing $\Omega(W_j^{1/19}/\text{poly}\log n)$ demand pairs of $\mathcal{M}^j$ in G_j; or (ii) a collection $\tilde{\mathcal{M}}^j \subseteq \mathcal{M}^j$ of demand pairs, with $|\tilde{\mathcal{M}}^j| \geq |\mathcal{M}^j|/2$, such that $\mathsf{OPT}(G_j, \tilde{\mathcal{M}}^j) \leq w^*|\tilde{\mathcal{M}}^j|/8$.

We say that instance $(G_j, \mathcal{M}^j)$ is a type-1 instance, if the first outcome happens for it, and we say that it is a type-2 instance otherwise. Let $I_1 = \left\{j \mid (G_j, \mathcal{M}^j) \text{ is a type-1 instance}\right\}$, and similarly, $I_2 = \left\{j \mid (G_j, \mathcal{M}^j) \text{ is a type-2 instance}\right\}$. We consider two cases. The first case happens when $\sum_{j \in I_1} W_j \geq W/2$, and the second case when $\sum_{j \in I_2} W_j \geq W/2$. In the second case, we let $\mathcal{M}' = \bigcup_{j \in I_2} \tilde{\mathcal{M}}^j$, and by the same reasoning as in Section 4, the following inequality, that is violated by the current LP-solution, is a valid constraint of (LP-flow2): $\sum_{(s_i,t_i) \in \mathcal{M}'} x'_i \leq w^*|\mathcal{M}'|/2$.

We now focus on Case 1, where the number of paths routed for each instance $(G_j, \mathcal{M}^j)$ with $j \in I_1$ is at least $|\mathcal{P}^j| = \Omega\left(\frac{W_j^{1/19}}{\text{poly}\log n}\right)$. Since $\sum_{j \in I_1} W_j \geq W/2 = \Omega\left(X^*/\log k\right)$, the total number of paths routed is:

$$\begin{aligned}\sum_{j \in I_1} |\mathcal{P}^j| &\geq \sum_{j \in I_1} \Omega\left(\frac{W_j^{1/19}}{\text{poly}\log n}\right) \\ &\geq \sum_{j \in I_1} \Omega\left(\frac{W_j}{W^{18/19} \cdot \text{poly}\log n}\right) \\ &= \Omega\left(\frac{W^{1/19}}{\text{poly}\log n}\right) = \Omega\left(\frac{(X^*)^{1/19}}{\text{poly}\log n}\right).\end{aligned}$$

9. REFERENCES

[1] Eyal Amir, Robert Krauthgamer, and Satish Rao. Constant factor approximation of vertex-cuts in planar graphs. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 90–99. ACM, 2003.

[2] Matthew Andrews. Approximation algorithms for the edge-disjoint paths problem via Raecke decompositions. In *Proceedings of IEEE FOCS*, pages 277–286, 2010.

[3] Matthew Andrews, Julia Chuzhoy, Venkatesan Guruswami, Sanjeev Khanna, Kunal Talwar, and Lisa Zhang. Inapproximability of edge-disjoint paths and low congestion routing on undirected graphs. *Combinatorica*, 30(5):485–520, 2010.

[4] Matthew Andrews and Lisa Zhang. Hardness of the undirected edge-disjoint paths problem. In *STOC*, pages 276–283. ACM, 2005.

[5] Yonatan Aumann and Yuval Rabani. Improved bounds for all optical routing. In *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, SODA '95, pages 567–576, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics.

[6] Chandra Chekuri and Julia Chuzhoy. Half-integral all-or-nothing flow. Unpublished Manuscript.

[7] Chandra Chekuri and Alina Ene. Poly-logarithmic approximation for maximum node disjoint paths with constant congestion. In *Proc. of ACM-SIAM SODA*, 2013.

[8] Chandra Chekuri, Sanjeev Khanna, and F Bruce Shepherd. Edge-disjoint paths in planar graphs. In

Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on, pages 71–80. IEEE, 2004.

[9] Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. Multicommodity flow, well-linked terminals, and routing problems. In *Proc. of ACM STOC*, pages 183–192, 2005.

[10] Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. An $O(\sqrt{n})$ approximation and integrality gap for disjoint paths and unsplittable flow. *Theory of Computing*, 2(1):137–146, 2006.

[11] Julia Chuzhoy. Routing in undirected graphs with constant congestion. In *Proc. of ACM STOC*, pages 855–874, 2012.

[12] Julia Chuzhoy and David H. K. Kim. On approximating node-disjoint paths in grids. In Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*, volume 40 of *LIPIcs*, pages 187–211. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[13] Julia Chuzhoy and Shi Li. A polylogarithimic approximation algorithm for edge-disjoint paths with congestion 2. In *Proc. of IEEE FOCS*, 2012.

[14] Shimon Even, Alon Itai, and Adi Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM J. Comput.*, 5(4):691–703, 1976.

[15] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[16] Ken-Ichi Kawarabayashi and Yusuke Kobayashi. An O(log n)-approximation algorithm for the edge-disjoint paths problem in Eulerian planar graphs. *ACM Trans. Algorithms*, 9(2):16:1–16:13, March 2013.

[17] Jon Kleinberg. An approximation algorithm for the disjoint paths problem in even-degree planar graphs. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '05, pages 627–636, Washington, DC, USA, 2005. IEEE Computer Society.

[18] Jon M. Kleinberg and Éva Tardos. Disjoint paths in densely embedded graphs. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 52–61, 1995.

[19] Jon M. Kleinberg and Éva Tardos. Approximations for the disjoint paths problem in high-diameter planar networks. *J. Comput. Syst. Sci.*, 57(1):61–73, 1998.

[20] Stavros G. Kolliopoulos and Clifford Stein. Approximating disjoint-path problems using packing integer programs. *Mathematical Programming*, 99:63–87, 2004.

[21] MR Kramer and Jan van Leeuwen. The complexity of wire-routing and finding minimum area layouts for arbitrary vlsi circuits. *Advances in computing research*, 2:129–146, 1984.

[22] James F. Lynch. The equivalence of theorem proving and the interconnection problem. *SIGDA Newsl.*, 5(3):31–36, September 1975.

[23] Harald Räcke. Minimizing congestion in general networks. In *Proc. of IEEE FOCS*, pages 43–52, 2002.

[24] Prabhakar Raghavan and Clark D. Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, December 1987.

[25] Satish Rao and Shuheng Zhou. Edge disjoint paths in moderately connected graphs. *SIAM J. Comput.*, 39(5):1856–1887, 2010.

[26] Heike Ripphausen-Lipa, Dorothea Wagner, and Karsten Weihe. Linear-time algorithms for disjoint two-face paths problems in planar graphs. *International Journal of Foundations of Computer Science*, 07(02):95–110, 1996.

[27] N. Robertson and P. D. Seymour. Outline of a disjoint paths algorithm. In *Paths, Flows and VLSI-Layout*. Springer-Verlag, 1990.

[28] Neil Robertson and Paul D Seymour. Graph minors. vi. disjoint paths across a disc. *Journal of Combinatorial Theory, Series B*, 41(1):115–138, 1986.

[29] Neil Robertson and Paul D Seymour. Graph minors. XIII. the disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995.

[30] Loïc Seguin-Charbonneau and F. Bruce Shepherd. Maximum edge-disjoint paths in planar graphs with congestion 2. In *Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 200–209, Washington, DC, USA, 2011. IEEE Computer Society.

[31] Hitoshi Suzuki, Takehiro Akama, and Takao Nishizeki. Finding steiner forests in planar graphs. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '90, pages 444–453, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics.