Routing in Undirected Graphs with Constant Congestion

Julia Chuzhoy*

November 3, 2011

Abstract

Given an undirected graph $G = (V, E)$, a collection $(s_1, t_1), \ldots, (s_k, t_k)$ of k demand pairs, and an integer c, the goal in the Edge Disjoint Paths with Congestion problem is to connect maximum possible number of the demand pairs by paths, so that the maximum load on any edge (called edge congestion) does not exceed c.

We show an efficient randomized algorithm to route $\Omega(\mathsf{OPT}/\operatorname{poly}\log k)$ demand pairs with congestion at most 14, where OPT is the maximum number of pairs that can be simultaneously routed on edge-disjoint paths. The best previous algorithm that routed $\Omega(\mathsf{OPT}/\operatorname{poly}\log n)$ pairs required congestion $\operatorname{poly}(\log\log n)$, and for the setting where the maximum allowed congestion is bounded by a constant c, the best previous algorithms could only guarantee the routing of $\mathsf{OPT}/n^{O(1/c)}$ pairs.

We also introduce a new type of vertex sparsifiers that we call *integral flow sparsifiers*, that approximately preserve both fractional and integral routings, and show an algorithm to construct such sparsifiers.

*Toyota Technological Institute, Chicago, IL 60637. Email: `cjulia@ttic.edu`. Supported in part by NSF CAREER grant CCF-0844872 and Sloan Research Fellowship.

1 Introduction

We study network routing problems in undirected graphs. In such problems, we are given an undirected n-vertex graph $G = (V, E)$, and a collection $\mathcal{M} = \{(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)\}$ of k *source-sink pairs*, that we also refer to as *demand pairs*. In order to route a pair (s_i, t_i), we need to select a path connecting s_i to t_i in graph G. Given a routing of any subset of the demand pairs, its *congestion* is the maximum load on any edge, that is, the maximum number of paths containing the same edge. In general, we would like to route as many demand pairs as possible, while minimizing the edge congestion. These two conflicting objectives naturally give rise to several basic optimization problems.

One of the central routing problems is Edge Disjoint Paths (EDP), where the goal is to route the maximum number of the demand pairs on edge-disjoint paths (that is, with congestion 1). Robertson and Seymour [RS90] have shown an efficient algorithm to solve this problem, when the number k of the demand pairs is bounded by a constant. However, for general values of k, it is NP-hard to even decide whether all pairs can be simultaneously routed on edge-disjoint paths [Kar72]. The best currently known approximation algorithm for the problem, due to Chekuri, Khanna and Shepherd [CKS06b], achieves an $O(\sqrt{n})$-approximation factor, while the best current hardness of approximation is $\Omega(\log^{1/2-\epsilon} n)$ for any constant ϵ, unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\text{poly}\log n})$ [AZ05, ACG+10]. We note that the standard multicommodity flow LP relaxation for EDP, that is commonly used in approximation algorithms for network routing problems, has an integrality gap of $\Omega(\sqrt{n})$ [CKS06b]. Interestingly, Rao and Zhou [RZ10] have shown a $\text{poly}\log n$-approximation algorithm for EDP on graphs where the value of the global minimum cut is $\Omega(\log^5 n)$, by rounding the same LP relaxation.

On the other extreme is the Congestion Minimization problem, where we need to route all source-sink pairs, while minimizing the edge congestion. The classical randomized rounding technique of Raghavan and Thompson [RT87] gives the best currently known approximation algorithm for this problem, whose approximation factor is $O(\log n/\log\log n)$. On the negative side, Andrews and Zhang [AZ07] show that the problem is hard to approximate to within a factor of $\Omega\left(\frac{\log\log n}{\log\log\log n}\right)$ unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\text{poly}\log n})$.

A problem that lies between these two extremes, and is a natural framework for studying the tradeoff between the number of pairs routed and the edge congestion is the Edge Disjoint Paths with Congestion problem (EDPwC). We say that an algorithm $\mathcal{A}$ achieves a factor α-approximation with congestion c for EDPwC, iff it routes at least OPT/α of the source-sink pairs, and the congestion of this routing is bounded by c, where OPT is the maximum number of demand pairs that can be simultaneously routed on edge-disjoint paths. In particular, a very interesting question is whether, by slightly relaxing the capacity constraints, and allowing a small edge congestion, we can significantly increase the number of pairs routed.

When the congestion c is allowed to be as high as $\Omega(\log n/\log\log n)$, the randomized rounding algorithm of Raghavan and Thompson [RT87] gives a constant factor approximation for EDPwC. For smaller values of c, until recently, only $O(n^{1/c})$-approximation algorithms have been known [AR01, BS00, KS04]. In a recent breakthrough, Andrews [And10] has shown a randomized algorithm to route $\Omega\left(\frac{\mathsf{OPT}}{\log^{61} n}\right)$ pairs with congestion $O((\log\log n)^6)$. In another recent result, Kawarabayashi and Kobayashi [KK11] have shown an algorithm that routes $\Omega\left(\frac{\mathsf{OPT}}{n^{3/7}}\right)$ pairs with congestion 2, thus improving the best previously known $O(\sqrt{n})$-approximation for $c = 2$.

In this paper we show an efficient randomized algorithm, that routes $\Omega\left(\frac{\mathsf{OPT}}{\log^{23.5} k\log\log k}\right)$ demand pairs with congestion at most 14. We note that on the negative side, Andrews et al. [ACG+10] have shown that for any constant ϵ, for any $1 \leq c \leq O\left(\frac{\log\log n}{\log\log\log n}\right)$, there is no $O\left((\log n)^{\frac{1-\epsilon}{c+1}}\right)$-approximation algorithm for EDPwC with congestion c, unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\text{poly}\log n})$. Therefore, the best approximation factor one may hope to achieve for EDPwC in the setting where the allowed congestion is bounded by a constant is polylogarithmic.

While our algorithm is guaranteed w.h.p. to route at least $\Omega(\mathsf{OPT}/\operatorname{poly}\log k)$ of the demand pairs with constant congestion, we have no control over the choice of the pairs that are routed. In some applications it may be useful to be able to choose the specific pairs for the algorithm to route beforehand. While we do not expect to be able to pre-select an arbitrary collection of the demand pairs to be routed, under some conditions we can still have some control over their selection. We show that if the graph G is well-linked for a given set $\mathcal{T}$ of terminals (see a formal definition below), then we can efficiently find a partition $\mathcal{G}$ of the terminals of $\mathcal{T}$ into groups of size $\operatorname{poly}\log k$, such that, if we are given **any** collection $\mathcal{M} = \{(s_1, t_1), \ldots, (s_r, t_r)\}$ of demand pairs, where for each group $U \in \mathcal{G}$, the terminals of U appear in at most one pair in $\mathcal{M}$, then there is an efficient randomized algorithm that w.h.p. routes all demand pairs in $\mathcal{M}$ with constant congestion.

We then turn to vertex flow sparsifiers. Given a graph G with a subset $\mathcal{T}$ of k vertices called terminals, and a set D of demands over the set $\mathcal{T}$, let $\eta(G, D)$ be the minimum congestion required to fractionally route the demands in D in graph G. We say that a graph H is a quality-q vertex flow sparsifier for $(G, \mathcal{T})$ iff $\mathcal{T} \subseteq V(H)$, and for any set D of demands over $\mathcal{T}$, $\eta(G, D) \leq \eta(H, D) \leq q\eta(G, D)$. Flow sparsifiers were first introduced by Moitra [Moi09] and Leighton and Moitra [LM10]. Their motivation was obtaining better approximation algorithms for combinatorial optimization problems, whose value only depends on the congestion $\eta(G, D)$ for various sets D of demands. The improvement is obtained by running the approximation algorithms on the sparsifier H instead of G, assuming that $|V(H)| << |V(G)|$. Several efficient algorithms are now known for constructing quality-$O(\log k/\log\log k)$ spasifiers H with $V(H) = \mathcal{T}$ [CLLM10, EGK+10, MM10]. However, such sparsifiers do not preserve *integral routings*. For example, if we were to solve the EDP problem, or some other routing problem on graph H, we are not guaranteed that we can transform this solution into an integral solution in graph G. This motivates our definition of *integral sparsifiers*, that approximately preserve both fractional and integral routings. Suppose we are given any n-vertex graph $G = (V, E)$ with a subset $\mathcal{T}$ of k vertices called terminals. We say that a graph H is a quality-(q_1, q_2)-integral flow sparsifier for G, iff (1) $\mathcal{T} \subseteq V(H)$; (2) for any set D of demands over $\mathcal{T}$, $\eta(H, D) \leq q_1\eta(G, D)$ (so in particular if we scale the demands in D down by factor q_1, we can route them fractionally in H with no congestion), and (3) given any integral routing $\mathcal{P}$ of any set $\mathcal{M}$ of pairs of terminals in graph H with congestion η, there is an efficient randomized algorithm to find an integral routing $\mathcal{P}'$ of $\mathcal{M}$ in G with congestion at most $q_2 \cdot \eta$. We show an efficient algorithm to construct integral sparsifiers H of quality (q_1, q_2) with $q_1 = \operatorname{poly}\log k$, $q_2 = 31$, and $|V(H)| = O(d)$, where d is the sum of degrees of all terminals.

Other related results EDP and its variants have been studied extensively, and better approximation algorithms are known for several special cases. Some examples include planar graphs [Fra85, KT95, Kle05, CKS05, CKS06a, KK10], trees [GVY93, CMS07], and expander graphs [LR99, BFU94, BFSU94, KR96, Fri00].

We note that routing problems are somewhat better understood in directed graphs. The EDP problem has $\tilde{O}\left(\min\left\{n^{2/3}, \sqrt{m}\right\}\right)$-approximation algorithms in directed graphs, where m is the number of graph edges [CK03, VV04, Kle96], and it is hard to approximate to within a factor of $\Omega\left(m^{1/2-\epsilon}\right)$ for any constant ϵ [GKR+99]. The randomized rounding technique of Raghavan and Thompson [RT87] gives an $O(\log n/\log\log n)$-approximation for directed Congestion Minimization, and the problem is hard to approximate to within a factor of $\Omega(\log n/\log\log n)$, unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\text{poly}\log n})$ [AZ08, CGKT07]. As for EDPwC, the randomized rounding technique gives an $O(cn^{1/c})$-approximation [KS04, Sri97] for any congestion bound c. On the other hand, for any $1 \leq c \leq O\left(\frac{\log n}{\log\log n}\right)$, there is no $n^{\Omega(1/c)}$-

approximation algorithm for the problem unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\text{poly}\log n})$ [CGKT07].

Our results and techniques. Our main result is summarized in the following theorem.

Theorem 1 *There is a randomized polynomial-time algorithm, that, given an undirected graph G and a set $\mathcal{M} = \{(s_1,t_1),\ldots,(s_k,t_k)\}$ of k demand pairs, w.h.p. finds a collection $\mathcal{P}$ of paths, connecting $\Omega\left(\frac{\mathsf{OPT}}{\log^{23.5} k \log\log k}\right)$ of the demand pairs with congestion at most 14, where OPT is the maximum number of the demand pairs that can be simultaneously routed on edge-disjoint paths in G.*

Our algorithm in fact routes $\Omega\left(\frac{\mathsf{OPT_{LP}}}{\log^{23.5} k \log\log k}\right)$ demand pairs, where $\mathsf{OPT_{LP}}$ is the value of the optimal solution to the standard multicommodity flow linear programming relaxation for the problem. Since the integrality gap of this LP relaxation is $\Omega(\sqrt{n})$ for EDP when no congestion is allowed, our result shows that the integrality gap improves from polynomial to polylogarithmic if we allow a congestion of 14.

A basic notion used throughout the paper is that of well-linkedness. Well-linkedness and its variations have been used extensively in previous work [CKS05, RZ10, And10]. We say that a graph $G = (V,E)$ is α-well-linked[1], iff for any partition (A,B) of V, $|E(A,B)| \geq \alpha \cdot \min\{|\mathcal{T} \cap A|, |\mathcal{T} \cap B|\}$.

Suppose we are given a graph $G = (V,E)$, a set $\mathcal{T} \subseteq V$ of vertices called terminals, a partition $\mathcal{G}$ of $\mathcal{T}$, and a collection $\mathcal{M} = \{(s_1,t_1),\ldots,(s_r,t_r)\}$ of demands pairs, where for each $1 \leq i \leq r$, $s_i, t_i \in \mathcal{T}$. We say that the demand set $\mathcal{M}$ is $(1,\mathcal{G})$-restricted, iff for every group $U \in \mathcal{G}$, at most one demand pair (s_i,t_i) contains a terminal of U (and only one terminal of U may participate in this pair). Our next theorem allows us to pre-select, to some extent, the demand pairs to be routed, if the set $\mathcal{T}$ of terminals is well-linked in G.

Theorem 2 *Suppose we are given an n-vertex graph $G = (V,E)$, a subset $\mathcal{T} \subseteq V$ of k_0 vertices called terminals, such that G is α_0-well-linked for $\mathcal{T}$, and an integer $c \geq 1$. Then we can efficiently find a partition $\mathcal{G}$ of the terminals in $\mathcal{T}$ into groups of size $O\left(\frac{(\log k_0)^{21+11/c}}{\alpha_0}\right)$, such that, given any set $\mathcal{M}$ of demand pairs over $\mathcal{T}$, where $\mathcal{M}$ is $(1,\mathcal{G})$-restricted, there is an efficient randomized algorithm that w.h.p. finds a routing of all pairs in $\mathcal{M}$ with congestion at most $14c+1$.*

In particular, we can achieve congestion 15 with group size $O(\log^{32} k_0)$, and if the group size is $O(\log^{22} k_0)$, then the congestion is 155. Finally, the next theorem provides an algorithm for constructing integral sparsifiers.

Theorem 3 *There is an efficient algorithm that constructs, for any graph G and a set $\mathcal{T}$ of k terminals, an integral sparsifier H of quality (q_1,q_2), with $q_1 = \text{poly}\log k$, $q_2 = 31$, and $|V(H)| = O(d)$, where d is the sum of degrees of all terminals.*

We now give an overview of our techniques and compare them to previous work. The starting point of the proof of Theorem 1 is the same as in the work of [CKS05, RZ10, And10]. We start with the standard LP-relaxation for the EDP problem on graph G, and we compute a partition of G into disjoint induced sub-graphs $G_1,\ldots,G_r$. For each $1 \leq i \leq r$, we compute a subset $\mathcal{M}_i \subseteq \mathcal{M}$ of demand pairs that are contained in G_i, such that G_i is well-linked for the corresponding set $\mathcal{T}_i$ of terminals, containing all vertices that participate in the pairs in $\mathcal{M}_i$, and moreover, $\sum_{i=1}^r |\mathcal{M}_i| \geq \Omega\left(\frac{|\mathcal{M}|}{\log^2 k}\right)$.

[1]Our definition of well-linkedness is similar to the notion of cut well-linkedness of [CKS05] (though we should say "set $\mathcal{T}$ of terminals is α-well linked in graph G" using their terminology).

An algorithm for efficiently computing such a decomposition was shown by Chekuri, Khanna and Shepherd [CKS05]. From now on, it is enough to find a good routing in each resulting sub-instance G_i separately. To simplify notation, let G denote any such sub-instance G_i, let $\mathcal{M}$ denote the set $\mathcal{M}_i$ of demand pairs, and let $\mathcal{T}$ denote the corresponding set $\mathcal{T}_i$ of terminals. Since graph G is well-linked for $\mathcal{T}$, it has good expansion properties with respect to $\mathcal{T}$. However, graph G may be far from being an expander, since it may contain many vertices besides the terminals. Intuitively, a natural approach is to embed an expander X, whose vertex set is $\mathcal{T}$, into the graph G. Each edge $e = (t_i, t_j)$ of the expander is mapped to a path P_e connecting t_i to t_j in G, and the congestion of the embedding is the maximum, over all edges $e' \in E(G)$, of the number of paths in $\{P_e \mid e \in E(X)\}$, containing e'. If we could find a low-congestion embedding of an expander X into G, then we could use existing algorithms for routing on expanders to find a low-congestion routing of a polylogarithmic fraction of the demand pairs in X, which in turn would give us a low-congestion routing of the same demand pairs in G. This is the approach that has been used by Rao and Zhou [RZ10] and by Andrews [And10]. A very useful tool in embedding an expander into any well-linked graph is the cut-matching game of Khandekar, Rao and Vazirani [KRV06]. In this game, we have two players: a cut player, who wants to construct an expander X, and a matching player, who tries to delay its construction. We start with X containing only the set $V(X)$ of $2N$ vertices and no edges. In each iteration i, the cut player computes a partition (A_i, B_i) of $V(X)$ with $|A_i| = |B_i| = N$, and the matching player computes a matching M_i between A_i and B_i. The edges of M_i are then added to X. Khandekar, Rao and Vazirani [KRV06] have shown that no matter what the matching player does, there is a strategy for the cut player (that we denote by $\mathcal{A}_{\mathrm{KRV}}$), such that after $O(\log^2 N)$ iterations, X becomes an expander. A natural approach to constructing an expander X and embedding it into the graph G using the cut-matching game, is the following. We use the algorithm $\mathcal{A}_{\mathrm{KRV}}$ for the cut player, while the matching player is simulated by finding appropriate flows in G. Specifically, we let $V(X) = \mathcal{T}$ be the set of vertices of X. If (A_i, B_i) is the bi-partition of $V(X)$ computed by the cut player, then we can try to send $|A_i| = |B_i|$ flow units from the terminals of A_i to the terminals of B_i in graph G, and use the resulting flow to define the matching M_i. This procedure can be used to both construct the expander X, and embed it into G. In fact, Khandekar, Rao and Vazirani use precisely this procedure in their algorithm for the sparsest cut problem.

One problem with this approach is that we need to compute $\Theta(\log^2 k)$ different flows in graph G, and together they may cause a poly-logarithmic congestion. Moreover, the partitions that the cut player computes depend on the matchings computed in previous iterations, so we cannot attempt to route all these flows simultaneously in graph G with low congestion. Rao and Zhou [RZ10] have proposed the following approach to overcome this difficulty. Let $\gamma = \Theta(\log^2 k)$ be the number iterations in the algorithm of [KRV06]. We can build γ graphs $G_1, \ldots, G_\gamma$, where for each $1 \leq i \leq \gamma$, $V(G_i) = V(G)$, and the sets $E(G_1), \ldots, E(G_\gamma)$ of edges form a partition of the edges in $E(G)$. If we can construct the family $G_1, \ldots, G_\gamma$ of graphs so that each graph G_i is still well-linked for the terminals, then we can now construct the expander X and embed it into G by using the cut-matching game of [KRV06], where in each iteration i, matching M_i is computed by finding a flow from A_i to B_i in graph G_i. Since the edges of each set M_i are embedded into distinct graphs G_i, the congestion does not accumulate, and we obtain a good embedding of X into G. In order to construct the graphs G_i, Rao and Zhou use a random procedure where each edge $e \in E$ is added to one of the graphs G_i uniformly at random. However, this procedure only works if the value of the global minimum cut in G is at least polylogarithmic. In order to overcome this difficulty, Andrews [And10] uses Raecke's tree decomposition technique [Räc02]. Roughly speaking, he decomposes the graph G into a collection $\mathcal{C}$ of disjoint clusters, where each cluster $C \in \mathcal{C}$ has some useful properties that allow us to find good routings across the cluster C efficiently. Moreover, if H is the graph obtained from G by contracting each cluster $C \in \mathcal{C}$ into a single vertex, then H is both well-linked for the terminals, and has a large global minimum cut, so we can use the algorithm of Rao and Zhou to complete the routing.

Our algorithm uses a slightly different way to embed an expander into G, somewhat similar to the one in [RZ10]. Specifically, each vertex $t \in V(X)$ is represented by a connected component C_t in graph G, that contains the terminal t. Each edge $e = (t, t') \in E(X)$ is represented by a path P_e connecting some vertex $v \in C_t$ to some vertex $v' \in C_{t'}$ in G. We ensure that each edge $e' \in E(G)$ only participates in a constant number of the components $\{C_t\}_{t \in V(X)}$ and paths $\{P_e\}_{e \in E(X)}$. Once we find such an embedding, we use **vertex-disjoint** routing in the expander X, that gives a low edge congestion routing in the original graph G.

A major point of our departure from previous work is in how the expander X is constructed and embedded into G. A central notion in our algorithm is that of a good family of vertex sets. Let $k' = k/\operatorname{poly}\log k$ be a parameter, where $k = |\mathcal{M}|$ is the number of the demand pairs. For any subset $S \subseteq V$ of vertices, let $\operatorname{out}(S)$ be the set of edges with exactly one endpoint in S. Given a subset $S \subseteq V$ of vertices of G, we say that S is α-well-linked, iff the graph $G[S]$ is α-well-linked for the set $\operatorname{out}(S)$ of edges. (More formally, subdivide every edge $e \in \operatorname{out}(S)$ with a new vertex t_e, and consider the sub-graph H_S of the resulting graph induced by $S \cup \mathcal{T}'$, where $\mathcal{T}' = \{t_e \mid e \in \operatorname{out}(S)\}$. We say that the set S is α-well-linked iff the graph H_S is α-well-linked for the set $\mathcal{T}'$ of terminals). Similarly, if we are given a subset $\Gamma \subseteq \operatorname{out}(S)$ of edges, we say that S is α-well-linked for Γ iff the graph $G[S]$ is α-well-linked for the set Γ of edges.

We say that a subset $S \subseteq G$ of vertices is a *good subset*, iff there is a collection $\Gamma \subseteq \operatorname{out}(S)$ of k' edges, such that S is α-well-linked for Γ (where $\alpha = 1/\operatorname{poly}\log k$), and moreover the edges in Γ can send $|\Gamma|$ flow units in G to the terminals in $\mathcal{T}$ with constant edge congestion. A family $\mathcal{F}$ of vertex subsets is a *good family* iff it contains γ mutually disjoint good vertex subsets $S_1, \ldots, S_\gamma$, where $\gamma = O(\log^2 k)$ is the parameter from the cut-matching game of [KRV06].

Suppose we find a good family $\mathcal{F} = \{S_1, \ldots, S_\gamma\}$ of vertex subsets. For each $1 \le j \le \gamma$, let $\Gamma_j \subseteq \operatorname{out}(S_j)$ be the corresponding subset Γ of edges. In order to construct the expander X, we select a subset $\mathcal{T}' = \{t_1, \ldots, t_{k'}\} \subseteq \mathcal{T}$ of k' terminals, and we let $V(X) = \mathcal{T}'$. For each $1 \le i \le k'$, we then construct a connected component C_i in graph G, that contains, for each $1 \le j \le \gamma$, a distinct edge $e_{i,j} \in \Gamma_j$, and also contains the terminal t_i. For each $1 \le j \le \gamma$, the edges $e_{1,j}, \ldots, e_{k',j}$ are all distinct, and we view the edge $e_{i,j}$ as the copy of terminal t_i for the set S_j. We also ensure that each edge of graph G only participates in a constant number of components $\{C_i\}_{i=1}^{k'}$. For each i, C_i is viewed as representing the vertex t_i of X in graph G. In order to construct the expander X, we use the cut-matching game of [KRV06], where in each iteration $1 \le j \le \gamma$, we use the sub-graph $G[S_j]$ to route some matching M_j between the copies of the terminals in A_j and B_j for the set S_j. This ensures that the congestion does not accumulate across different iterations. Finally, we show an efficient algorithm for finding a good family $\mathcal{F}$ of vertex subsets.

Organization. Most of this paper is devoted to the proof of our main result, Theorem 1. We start with preliminaries in Section 2, and provide the proof in Section 3. For convenience, a list of parameters is provided in Section A of the Appendix. The proofs of Theorems 2 and 3 appear in Sections 4 and 5 respectively.

2 Preliminaries and Notation

In this section we provide notation and basic results that we use to prove Theorem 1. We assume that we are given an undirected n-vertex graph $G = (V, E)$, and a set $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of k source-sink pairs, that we also refer to as demand pairs. We denote by $\mathcal{T}$ the set of vertices that participate in pairs in $\mathcal{M}$, and we call them *terminals*. Let OPT denote the maximum number of demand pairs that can be simultaneously routed via edge-disjoint paths. Our goal is to connect $\Omega(\mathsf{OPT}/\operatorname{poly}\log k)$ distinct pairs in $\mathcal{M}$ with paths that cause congestion at most 14.

We assume w.l.o.g. that every terminal in $\mathcal{T}$ participates in exactly one source-sink pair. Otherwise, if a terminal $v \in \mathcal{T}$ participates in $r > 1$ source-sink pairs, we can add r new terminals $t_1(v), \ldots, t_r(v)$, connect each of them to v with an edge, and use a distinct terminal in $\{t_1(v), \ldots, t_r(v)\}$ for each source-sink pair in which v participates. We also assume w.l.o.g. that the maximum vertex degree in G is 4, and that the degree of every terminal is 1. In order to achieve this, we perform the following simple transformation of graph G. If v is a terminal, whose degree is greater than 1, then we add a new vertex u to graph G that connects to v with an edge, and becomes a terminal instead of v. Next, we process the non-terminal vertices one-by-one. Let v be any such vertex, and assume that the degree of v is $d > 4$. Let $u_1, \ldots, u_d \in V$ be the neighbors of v. We replace v with a $d \times d$ grid Z_v, and denote by $u'_1, \ldots, u'_d$ the vertices in the first row of Z_v. For each $1 \leq i \leq d$, we add an edge (u_i, u'_i). It is easy to verify that any solution to the EDP problem in the original graph can be transformed into a feasible routing of the same value and no congestion in the new graph, and any routing in the new graph with congestion η can be transformed into a routing in the original graph with the same congestion. Therefore, we assume from now on that the maximum vertex degree in G is 4, the degree of every terminal is 1, and every terminal participates in one source-sink pair.

For any subset $S \subseteq V$ of vertices, we denote by $\operatorname{out}_G(S) = E_G(S, V \setminus S)$, and by $E_G(S)$ the subset of edges with both endpoints in S. When clear from context, we omit the subscript G. Throughout the paper, we say that a random event succeeds w.h.p., if the probability of success is $(1 - 1/\operatorname{poly}(n))$.

Let $\mathcal{P}$ be any collection of paths in graph G. We say that paths in $\mathcal{P}$ cause congestion η in G, iff for every edge $e \in E$ at most η paths in $\mathcal{P}$ contain e. Assume that we are given a subset $S \subseteq V$ of vertices and a subset $E' \subseteq E$ of edges of G. We say that a collection $\mathcal{P}$ of paths connects the vertices of S to the edges of E' with congestion η, and denote $\mathcal{P} : S \leadsto_\eta E'$, iff $\mathcal{P} = \{P_v \mid v \in S\}$, where path P_v has v as its first vertex and some edge of E' as its last edge, and $\mathcal{P}$ causes congestion at most η in G. In particular, each edge in E' serves as the last edge on at most η paths in $\mathcal{P}$. Similarly, given two subsets S, S' of vertices, if $\mathcal{P}$ is a collection of paths, connecting every vertex of S to some vertex of S' with overall congestion at most η, then we denote this by $\mathcal{P} : S \leadsto_\eta S'$. Finally, if $|S| = |S'| = |\mathcal{P}|$, and each path in $\mathcal{P}$ connects a distinct vertex of S to a distinct vertex of S', then we denote this by $\mathcal{P} : S \overset{1:1}{\leadsto}_\eta S'$. Similarly, we say that a flow F connects the vertices of S to the edges of E' with congestion η, and denote $F : S \leadsto_\eta E'$, iff each vertex $v \in S$ sends one flow unit to the edges in E', and the flow F causes congestion at most η in G. Notice that each flow-path in F starts at a vertex of S and terminates at some edge $e \in E'$. We view edge e as part of the flow-path, so in particular each edge in E' receives at most η flow units. Notice that from the integrality of flow, there is a flow $F : S \leadsto_\eta E'$ iff there is a collection $\mathcal{P} : S \leadsto_\eta E'$ of paths. We define flows and paths between subsets of edges similarly.

We will often be interested in a scenario where we are given a subset S of vertices and two subsets $E_1, E_2 \subseteq \operatorname{out}(S)$ of edges. We say that the flow $F : E_1 \leadsto_\eta E_2$ is *contained in* S iff every flow-path is completely contained in $G[S]$, except for its first and last edges, that belong to $\operatorname{out}(S)$. Similarly, we say that a set $\mathcal{P} : E_1 \leadsto_\eta E_2$ of paths is contained in S iff all inner edges on every path of $\mathcal{P}$ belong to $G[S]$.

Given a graph $G = (V, E)$, and a set $\mathcal{T} \subseteq V$ of terminals, a set D of demands is a function $D : \mathcal{T} \times \mathcal{T} \to \mathbb{R}^+$, that specifies, for every unordered pair $t, t' \in \mathcal{T}$, a demand $D_{t,t'}$. We say that the set D of demands is γ-restricted, iff for each $t \in \mathcal{T}$, the total demand $\sum_{t' \in \mathcal{T}} D_{t,t'} \leq \gamma$. Given a partition $\mathcal{G}$ of the set $\mathcal{T}$ of terminals, we say that the set D of demands is $(\gamma, \mathcal{G})$-restricted, iff for each $U \in \mathcal{G}$, the total demand $\sum_{t \in U} \sum_{t' \in \mathcal{T}} D_{t,t'} \leq \gamma$. We say that the set D of demands is *integral* iff $D_{t,t'}$ is integral for each $t, t' \in \mathcal{T}$.

Given any set D of demands, a *fractional routing* of D is a flow F, where every unordered pair $t, t' \in \mathcal{T}$ sends $D_{t,t'}$ flow units to each other. Given an integral set D of demands, an *integral* routing of D is a collection $\mathcal{P}$ of paths, where for each unordered pair $(t, t') \in \mathcal{T}$, there are $D_{t,t'}$ paths connecting t to t' in $\mathcal{P}$. The congestion of this routing is the congestion caused by the set $\mathcal{P}$ of paths in G. Observe that any matching $\mathcal{M}$ on the set $\mathcal{T}$ of terminals defines a set D of demands where $D_{t,t'} = 1$ for $(t, t') \in \mathcal{M}$ and $D_{t,t'} = 0$ otherwise. We do not distinguish between the matching $\mathcal{M}$ and the set D of demands.

Sparsest Cut and the Flow-Cut Gap. Suppose we are given a graph $G = (V, E)$, with non-negative weights w_v on vertices $v \in V$, and a subset $\mathcal{T} \subseteq V$ of k terminals, such that for all $v \notin \mathcal{T}$, $w_v = 0$. For any subset $S \subseteq V$ of vertices, let $w(S) = \sum_{v \in S} w(v)$. The sparsity of a cut $(S, \overline{S})$ in G is $\Phi(S) = \frac{|E(S,\overline{S})|}{w(S)\cdot w(\overline{S})}$, and the value of the sparsest cut in G is defined to be: $\Phi(G) = \min_{S \subset V} \{\Phi(S)\}$. In the sparsest cut problem, the input is a graph G with non-negative vertex weights, and the goal is to find a cut of minimum sparsity. Arora, Rao and Vazirani [ARV09] have shown an $O(\sqrt{\log k})$-approximation algorithm for the sparsest cut problem. We will often work with a special case of the sparsest cut problem, where for each $t \in \mathcal{T}$, $w_t = 1$.

A problem dual to sparsest cut is the maximum concurrent flow problem. For the case where the weights of all terminals are unit, the goal in the maximum concurrent flow problem is to find the maximum value λ, such that every pair of terminals can send λ flow units to each other simultaneously with no congestion. The flow-cut gap is the maximum ratio, in any graph, between the value of the minimum sparsest cut and the maximum concurrent flow. The value of the flow-cut gap in undirected graphs, that we denote by $\beta(k)$ throughout the paper, is $\Theta(\log k)$ [LR99, GVY95, LLR94, AR98]. Therefore, if $\Phi(G) = \alpha$, then every pair of terminals can send $\alpha/\beta(k)$ flow units to each other with no congestion.

We will use a sightly different, but also standard, and roughly equivalent, definition of sparsity. Given any partition $(S, \overline{S})$ of V, the *sparsity* of the cut $(S, \overline{S})$ is $\Psi(S, \overline{S}) = \frac{|E(S,\overline{S})|}{\min\left\{w(S), w(\overline{S})\right\}}$. We then denote: $\Psi(G) = \min_{S \subset V} \left\{\Psi(S, \overline{S})\right\}$. It is easy to see that $2\Psi(G)/k \geq \Phi(G) \geq \Psi(G)/k$. Therefore, if $\Psi(G) = \alpha$, then $\Phi(G) \geq \alpha/k$, and every pair of terminals can send $\frac{\alpha}{k\beta(k)}$ flow units to each other with no congestion. Equivalently, every pair of terminals can send $1/k$ flow units to each other with congestion at most $\beta(k)/\alpha$. Moreover, any matching on the set $\mathcal{T}$ of terminals can be fractionally routed with congestion at most $2\beta(k)/\alpha$. In the rest of the paper, we will use the latter definition of sparsity, and we will use the term cut sparsity and the value of sparsest cut to denote $\Psi(S, \overline{S})$ and $\Psi(G)$ respectively. The algorithm of [ARV09] can still be used to obtain a cut of sparsity at most $O(\sqrt{\log k}) \cdot \Psi(G)$ in G. We denote by $\mathcal{A}_{\text{ARV}}$ this algorithm and by $\alpha_{\text{ARV}}(k) = O(\sqrt{\log k})$ its approximation factor.

Routing on Expanders. We say that a graph $G = (V, E)$ is an α-expander, iff $\min_{\substack{S \subseteq V:\\ |S| \leq |V|/2}} \left\{\frac{|E(S,\overline{S})|}{|S|}\right\} \geq \alpha$. There are many algorithms for routing on expanders, e.g. [LR99, BFU94, BFSU94, KR96, Fri00], which give different types of guarantees. For example, Frieze [Fri00] has shown that if G is an r-regular graph (where r is a constant) with strong enough expansion properties, then there is an efficient randomized algorithm for routing *any* matching on any subset of $\Omega(n/\log n)$ of its vertices via edge-disjoint paths. We need a slightly different type of guarantee: the routing should be on **vertex-disjoint** paths, and the graph degree may be super-constant (but still bounded). Rao and Zhou [RZ10] give such an algorithm, which is summarized in the next theorem. For completeness, we provide a proof sketch in the Appendix.

Theorem 4 (Theorem 7.1 in [RZ10]) *Let $G = (V, E)$ be any n-vertex d-regular α-expander, for $\alpha = 1/2$. Assume further that n is even, and that the vertices of G are partitioned into $n/2$ dis-*

joint demand pairs $\mathcal{M} = \{(s_1,t_1),\ldots,(s_{n/2},t_{n/2})\}$. *Then there is an efficient algorithm that routes* $\Omega\left(\frac{n}{\log n \cdot d^2}\right)$ *of the demand pairs on vertex-disjoint paths in* G.

The Cut-Matching Game. We use the cut-matching game of Khandekar, Rao and Vazirani [KRV06]. In this game, we are given a set V of N vertices, where N is even, and two players: a cut player, whose goal is to construct an expander X on the set V of vertices, and a matching player, whose goal is to delay its construction. The game is played in iterations. We start with the graph X containing the set V of vertices, and no edges. In each iteration j, the cut player computes a bi-partition (A_j,B_j) of V into two equal-sized sets, and the matching player returns some perfect matching M_j between the two sets. The edges of M_j are then added to X. The following theorem was proved in [KRV06].

Theorem 5 ([KRV06]) *There is a probabilistic algorithm for the cut player, such that, no matter how the matching player plays, after* $\gamma_{\mathrm{KRV}}(N) = O(\log^2 N)$ *iterations, graph* X *is a* $\frac{1}{2}$*-expander w.h.p.*

Well-Linked Decompositions. Well-linked decompositions have been used extensively in algorithms for network routing, e.g. in [Räc02, CKS04, CKS05, RZ10, And10]. We define a specific type of well-linkedness that our algorithm uses and give an algorithm for computing the corresponding well-linked decomposition.

Definition 1 *Given a graph* G, *a subset* S *of its vertices, and a parameter* $\alpha > 0$, *we say that* S *is* α*-well-linked, iff for any partition* (A,B) *of* S, *if we denote by* $T_A = \mathrm{out}(A) \cap \mathrm{out}(S)$, *and by* $T_B = \mathrm{out}(B) \cap \mathrm{out}(S)$, *then* $|E(A,B)| \geq \alpha \cdot \min\{|T_A|,|T_B|\}$.

We also need a more general notion of well-linkedness that we define below. Intuitively, it handles subsets S of vertices, where $|\mathrm{out}(S)|$ may be large, but we only need to route small amounts of flow across S.

Definition 2 *Let* S *be any subset of vertices of a graph* G. *For any integer* $k > 0$ *and for any* $0 < \alpha < 1$, *we say that set* S *is* (k,α)*-well-linked iff for any pair* $T_1, T_2 \subseteq \mathrm{out}(S)$ *of disjoint subsets of edges, with* $|T_1| + |T_2| \leq k$, *for any partition* (X,Y) *of* S *with* $T_1 \subseteq \mathrm{out}(X)$ *and* $T_2 \subseteq \mathrm{out}(Y)$, $|E_G(X,Y)| \geq \alpha \cdot \min\{|T_1|,|T_2|\}$.

Note that if $|\mathrm{out}(S)| \leq k$, then set S is (k,α)-well-linked iff it is α-well-linked, and the two definitions of well-linkedness become equivalent. Notice also that if S is (k,α)-well-linked, then for any subset $T \subseteq \mathrm{out}(S)$ of at most k edges, any matching on T can be fractionally routed inside S with congestion at most $2\beta(k)/\alpha$. This is since we can set up an instance of the sparsest cut problem on graph $G[S] \cup T$, where the edges of T serve as terminals. Since S is (k,α)-well-linked, the value of the sparsest cut is at least α, and so any matching on T can be routed with congestion at most $2\beta(k)/\alpha$.

Assume now that S is not (k,α)-well-linked. Then there must be a partition (X,Y) of S, and two subsets $T_1 \subseteq \mathrm{out}(X) \cap \mathrm{out}(S)$, $T_2 \subseteq \mathrm{out}(Y) \cap \mathrm{out}(S)$ with $|T_1| + |T_2| \leq k$, such that $|E(X,Y)| < \alpha \cdot \min\{|T_1|,|T_2|\}$. We say that (X,Y) is a (k,α)-*violating cut* for S.

Given a subset S of vertices of G, we would like to find a partition $\mathcal{W}$ of S, such that every set in $W \in \mathcal{W}$ is (k,α)-well-linked. We could do so using the standard well-linked decomposition procedures, for example similar to those used in [Räc02, CKS05]. However, in order to do so, we need to be able to check whether a given subset W of vertices is (k,α)-well-linked, and if not, find a (k,α)-violating cut efficiently. We do not know how to do this, even approximately. Therefore, we will assume for now that we are given an oracle that finds a (k,α)-violating cut in a given subset of vertices, if such a cut exists.

We describe the decomposition procedure and bound the number of edges $\sum_{W \in \mathcal{W}} |\operatorname{out}(W)|$ in the resulting decomposition. When we use this decomposition later in the algorithm, we will be interested in routing small amounts of flow (up to k) across the clusters of the decomposition. Whenever we will be unable to route this flow, we will naturally obtain a (k, α)-violating cut. Therefore, our algorithm itself will serve as an oracle to the decomposition procedure. We note that in the final decomposition $\mathcal{W}$, not all sets $W \in \mathcal{W}$ may be (k, α)-well-linked, but we will be able to route the flow that we need to route across these clusters, and this is sufficient for us. We now describe the oracle-based decomposition procedure and analyze it.

We are given as input a subset S of vertices of G, an integer k, and a parameter $0 < \alpha < 1$. Throughout the decomposition procedure, we maintain a partition $\mathcal{W}$ of S, and at the beginning, $\mathcal{W} = \{S\}$. The algorithm proceeds as follows. As long as not all sets in $\mathcal{W}$ are (k, α)-well-linked, our oracle computes a (k, α)-violating partition (X, Y) of one of the sets $W \in \mathcal{W}$. We then remove W from $\mathcal{W}$ and add X and Y to $\mathcal{W}$ instead. In the next theorem, we bound $\sum_{W \in \mathcal{W}} |\operatorname{out}(W)|$.

Theorem 6 *Let $k > 8$, and denote $\gamma = \gamma_{\mathrm{KRV}}(k) = \Theta(\log^2 k)$. Let $\alpha(k) = \frac{1}{2^{11} \cdot \gamma \cdot \log k}$, and let $\mathcal{W}$ be any partition of S produced over the course of the above algorithm. Then $\sum_{W \in \mathcal{W}} |\operatorname{out}(W)| \leq |\operatorname{out}(S)| \left(1 + \frac{1}{64\gamma}\right)$.*

We emphasize that the bound on $\sum_{W \in \mathcal{W}} |\operatorname{out}(W)|$ holds for any partition produced over the course of the algorithm, and not just the final partition.

Proof: The proof uses a standard charging scheme. For simplicity, we denote $\alpha = \alpha(k)$. Consider some iteration of the algorithm, and suppose the oracle has found a (k, α)-violating partition (X, Y) of some set W in the current partition. Let $T_X = \operatorname{out}(X) \cap \operatorname{out}(W)$, $T_Y = \operatorname{out}(Y) \cap \operatorname{out}(W)$, and assume w.l.o.g. that $|T_X| \leq |T_Y|$ (note that it is possible that $|T_X| > k$). We charge the edges of T_X evenly for the edges in $E(X, Y)$. Specifically, if $|T_X| \geq k/2$, then $|E(X, Y)| \leq \alpha k/2$ must hold, and the charge to each edge in T_X is at most $\frac{\alpha k}{2|T_X|} \leq \frac{\alpha k}{|\operatorname{out}(X)|}$, since $|\operatorname{out}(X)| = |T_X| + |E(X, Y)| \leq 2|T_X|$. Otherwise, $|E(X, Y)| \leq \alpha \cdot |T_X|$, and the charge to each edge of T_X is at most α. In any case, $|\operatorname{out}(X)| = |T_X| + |E(X, Y)| < 2|\operatorname{out}(W)|/3$, and $|\operatorname{out}(Y)| \leq |\operatorname{out}(W)|$.

Consider some edge $e = (u, v) \in \bigcup_{W \in \mathcal{W}} \operatorname{out}(W)$. We analyze the charge to edge e. We first bound the charge via the vertex u. Let $i_1 \leq i_2 \leq \cdots \leq i_\ell$ be the iterations of the decomposition procedure in which e was charged via vertex u, and for each $1 \leq j \leq \ell$, let $z_j = |\operatorname{out}(W)|$, where W is the cluster to which u belonged at the end of iteration i_j. Note that for each $1 < j \leq \ell$, $z_j < 2z_{j-1}/3$. Let j^* be the largest index for which $z_{j^*} > k/2$. Then the total charge to e via u in iterations $i_1, \ldots, i_{j^*}$ is at most:

$$\frac{\alpha k}{z_1} + \frac{\alpha k}{z_2} + \cdots + \frac{\alpha k}{z_{j^*}} \leq \frac{\alpha k}{z_{j^*}} \left(1 + (2/3) + (2/3)^2 + \cdots + (2/3)^{j^*-1}\right) < \frac{3\alpha k}{z_{j^*}} \leq 6\alpha$$

In each subsequent iteration, the charge to edge e was at most α, and the number of such iterations is bounded by $2 \log k$. So the charge to edge e via vertex u is at most $6\alpha + 2\alpha \log k < 4\alpha \log k$, and the total charge to edge e is at most $8\alpha \log k \leq \frac{1}{2^8 \gamma}$. This however only accounts for the *direct* charge. For example, some edge $e' \notin \operatorname{out}(S)$, that was first charged to the edges in $\operatorname{out}(S)$, can in turn be charged for some other edges. We call such charging *indirect*. If we sum up the indirect charge for every edge $e \in \operatorname{out}(S)$, we obtain a geometric series, and so the total direct and indirect amount charged to every edge $e \in \operatorname{out}(S)$ is at most $\frac{1}{128\gamma}$. We conclude that $\sum_{W \in \mathcal{W}} |\operatorname{out}(W)| \leq |S| \left(1 + \frac{1}{64\gamma}\right)$. (The additional factor of 2 is due to the fact that each edge of the partition is counted twice in $\sum_{W \in \mathcal{W}} |\operatorname{out}(W)|$ - once for each its endpoint). □

Let $\alpha_{\text{WL}}(k) = \alpha(k)/\alpha_{\text{ARV}}(k) = \Omega(1/(\log^{3.5} k))$. If $|\text{out}(S)| \leq k$, then we can obtain a $(k, \alpha_{\text{WL}}(k))$-well-linked decomposition of S efficiently, by using the algorithm $\mathcal{A}_{\text{ARV}}$ for Sparsest Cut as our oracle: In each iteration, for each $W \in \mathcal{W}$, we apply the algorithm $\mathcal{A}_{\text{ARV}}$ to the corresponding instance of the sparsest cut problem (where the edges of $\text{out}(W)$ are viewed as terminals). If algorithm $\mathcal{A}_{\text{ARV}}$ returns a $(k, \alpha(k))$-violating cut (X, Y) for any set $W \in \mathcal{W}$, then we can proceed with the decomposition procedure as before. Otherwise, we are guaranteed that each set $W \in \mathcal{W}$ is $\alpha_{\text{WL}}(k)$-well-linked. We therefore have the following corollary.

Corollary 1 *Let S be any subset of vertices of G, such that $|\text{out}(S)| \leq k$. Then we can efficiently find a partition $\mathcal{W}$ of S, such that for each $W \in \mathcal{W}$, $|\text{out}(W)| \leq k$, and it is $\alpha_{\text{WL}}(k) = \frac{1}{2^{11} \cdot \alpha_{\text{ARV}}(k) \cdot \gamma_{\text{KRV}}(k) \cdot \log k} = \Omega(1/(\log^{3.5} k))$-well-linked. Moreover, $\sum_{W \in \mathcal{W}} |\text{out}(W)| \leq |\text{out}(S)| \left(1 + \frac{1}{64 \gamma_{\text{KRV}}(k)}\right)$.*

This finishes the description of the well-linked decomposition procedure. Throughout the paper, we use $\alpha(k) = \frac{1}{2^{11} \cdot \gamma_{\text{KRV}}(k) \cdot \log k}$ to denote the parameter from Theorem 6, and $\alpha_{\text{WL}}(k) = \alpha(k)/\alpha_{\text{ARV}}(k)$ to denote the parameter from Corollary 1.

The Grouping Technique. The grouping technique was first introduced by Chekuri, Khanna and Shepherd [CKS04], and has since been widely used in algorithms for network routing [CKS05, RZ10, And10], to boost network connectivity and well-linkedness parameters. We summarize it in the following theorem, whose proof appears in the Appendix for completeness.

Theorem 7 *Suppose we are given a graph $G = (V, E)$, with weights $w(v)$ on vertices $v \in V$, and a parameter p. Assume further that for each $v \in V$, $0 \leq w(v) \leq p$. Then we can find a partition $\mathcal{G}$ of the vertices in V, and for each group $U \in \mathcal{G}$, find a tree $T_U \subseteq G$ containing all vertices of U, such that the trees $\{T_U\}_{U \in \mathcal{G}}$ are edge-disjoint, and for each $U \in \mathcal{G}$, $p \leq \sum_{v \in U} w(v) \leq 3p$.*

We will sometimes use the grouping theorem in slightly different settings. The first such setting is when we are given a subset $\mathcal{T} \subseteq V$ of vertices called terminals, and we would like to group them into groups of cardinality at least p and at most $3p$. In this case we will think of all non-terminal vertices as having weight 0, and terminal vertices as having weight 1. Instead of finding a partition $\mathcal{G}$ of all vertices, we will be looking for a partition $\mathcal{G}'$ of the set $\mathcal{T}$ of terminals. This partition is obtained from $\mathcal{G}$ by ignoring the non-terminal vertices. Another setting is when we are given a subset $E' \subseteq E$ of edges, and we would like to find a grouping $\mathcal{G}$ of these edges into groups of at least p and at most $3p$ edges. As before, we would also like to find, for each group $U \in \mathcal{G}$, a tree T_U containing all edges in U, and we require that the trees $\{T_U\}_{U \in \mathcal{G}}$ are edge-disjoint. This setting can be reduced to the previous one, by sub-dividing each edge $e \in E'$ by a terminal vertex. It is easy to verify that Theorem 7 can be applied in this setting as well.

3 The Algorithm

This section is dedicated to proving Theorem 1. Our starting point is similar to that used in previous work on the problem [CKS04, CKS05, RZ10, And10]: namely, we use the standard multicommodity flow LP-relaxation for the EDP problem to partition our graph into several disjoint sub-graphs, that are well-linked for their respective sets of terminals, and solve the problem separately on each such sub-graph. Recall that the standard LP-relaxation for EDP is defined as follows. For each $1 \leq i \leq k$, we have an indicator variable x_i for whether or not we route the pair (s_i, t_i). Let $\mathcal{P}_i$ denote the set of all paths connecting s_i to t_i in G. The LP relaxation is defined as follows.

$$
\begin{array}{rll}
(LP) & \max & \sum_{i=1}^{k} x_i \\
\text{s.t.} & \sum_{P\in\mathcal{P}_i} f(P)\geq x_i & \forall 1\leq i\leq k \\
& \sum_{P:e\in P} f(P)\leq 1 & \forall e\in E \\
& 0\leq x_i\leq 1 & \forall 1\leq i\leq k \\
& f(P)\geq 0 & \forall 1\leq i\leq k, \forall P\in\mathcal{P}_i
\end{array}
$$

While this LP has exponentially many variables, it can be efficiently solved using standard techniques, e.g. by using an equivalent polynomial-size LP formulation. Throughout the rest of the paper, we denote by OPT the value of the optimal solution to the LP. Clearly, the value of the optimal solution to the EDP problem instance is at most OPT. We need the following definition.

Definition 3 *Given a graph $G=(V,E)$, and a subset $\mathcal{T}\subseteq V$ of vertices called terminals, we say that $\mathcal{T}$ is* flow-well-linked *in G, iff any matching $\mathcal{M}$ on $\mathcal{T}$ can be fractionally routed with congestion at most 2 in G.*

The next theorem follows from the work of Chekuri, Khanna and Shepherd [CKS04, CKS05], and we provide a short proof sketch in the Appendix for completeness.

Theorem 8 *Suppose we are given a graph $G=(V,E)$ and a set $\mathcal{M}$ of k source-sink pairs in G. Then we can efficiently partition G into a collection $G_1,\ldots,G_\ell$ of vertex-disjoint induced sub-graphs, and compute, for each $1\leq i\leq \ell$, a collection $\mathcal{M}_i\subseteq\mathcal{M}$ of source-sink pairs contained in G_i, such that $\sum_{i=1}^{\ell}|\mathcal{M}_i|=\Omega(\mathsf{OPT}/\log^2 k)$, and moreover, if for each $1\leq i\leq\ell$, $\mathcal{T}_i$ denotes the set of terminals participating in pairs in $\mathcal{M}_i$, then $\mathcal{T}_i\subseteq V(G_i)$, and G_i is flow-well-linked for $\mathcal{T}_i$.*

We now proceed to solve the problem on each one of the graphs G_i separately. In order to simplify the notation, we denote the graph G_i by G, the set $\mathcal{M}_i$ of the source-sink pairs by $\mathcal{M}$, and the set of terminals by $\mathcal{T}$. For simplicity, we denote $|\mathcal{M}|=k$. Recall that G is flow-well-linked for $\mathcal{T}$, the degree of every terminal in $\mathcal{T}$ is 1, and the maximum vertex degree in G is at most 4. It is now enough to prove that we can route $\Omega\left(\frac{k}{\log^{21.5}k\log\log k}\right)$ demand pairs in $\mathcal{M}$ with congestion at most 14. We also assume that $k>k_0$, where k_0 is a large enough constant: otherwise, we can simply pick any source-sink pair $(s,t)\in\mathcal{M}$, connect it with any path P and output this as a solution. In particular, we will assume that $k>\log^{24}k$, and $\gamma_{\text{KRV}}(k)=\Theta(\log^2 k)>20$.

Legal Contracted Graph. Let $\gamma=\gamma_{\text{KRV}}(k)=\Theta(\log^2 k)$. We use a parameter $k_1=\frac{k}{192\gamma^3\log\gamma}=\Omega\left(\frac{k}{\log^6 k\log\log k}\right)$. We maintain, throughout the algorithm, a graph G', obtained from G by contracting some subsets of non-terminal vertices of G. We say that G' is a *legal contracted graph* for G, iff the following conditions hold:

- The set $V(G')$ is partitioned into two subsets, $V_1\subseteq V(G)$ containing the original vertices of G, and $V_2=V(G')\setminus V_1$, containing super-nodes v_C, for $C\subseteq V(G)$. The subsets V_1 and $\{C\}_{v_C\in V_2}$ of vertices of G are all pairwise disjoint, and $\mathcal{T}\subseteq V_1$.
- Graph G' can be obtained from graph G by contracting each cluster in set $\{C\mid v_C\in V_2\}$ into the super-node v_C (we delete all self-loops, but we do not delete parallel edges).

- For each super-node $v_C \in V_2$, $|\mathrm{out}_G(C)| \leq k_1$, and the set C is $\alpha_{\mathrm{WL}}(k)$-well-linked in graph G.

Notice that graph G' may contain parallel edges, and it remains flow-well-linked for the set $\mathcal{T}$ of terminals. Also, since the maximum vertex degree in G is 4, the maximum vertex degree in G' is at most k_1, and every terminal has degree 1. Every edge in graph G' corresponds to some edge in the original graph G, and we will not distinguish between them. In particular, for every vertex subset $S' \subseteq V(G')$, if $S \subseteq V(G)$ is the corresponding subset of vertices in G, where every super-node $v_C \in S' \cap V_2$ is replaced by the vertices of C, then there is a one-to-one mapping between $\mathrm{out}_{G'}(S')$ and $\mathrm{out}_G(S)$, and we will identify the edges in these two sets, that is, $\mathrm{out}_{G'}(S') = \mathrm{out}_G(S)$. We need the following simple claim.

Claim 1 *If G' is a legal contracted graph for G, then $G' \setminus \mathcal{T}$ contains at least $k/6$ edges.*

Proof: For each terminal $t \in \mathcal{T}$, let e_t be the unique edge adjacent to t in G', and let u_t be the other endpoint of e_t. We partition the terminals in $\mathcal{T}$ into groups, where two terminals t, t' belong to the same group iff $u_t = u_{t'}$. Let $\mathcal{G}$ be the resulting partition of the terminals. Since the degree of every vertex in G' is at most k_1, each group $U \in \mathcal{G}$ contains at most k_1 terminals. Next, we partition the terminals in $\mathcal{T}$ into two subsets X, Y, where $|X|, |Y| \geq k/3$, and for each group $U \in \mathcal{G}$, either $U \subseteq X$, or $U \subseteq Y$ holds. It is possible to find such a partition by greedily processing each group $U \in \mathcal{G}$, and adding all terminals of U to one of the subsets X or Y, that currently contains fewer terminals. Finally, we remove terminals from set X until $|X| = k/3$, and we do the same for Y. Since graph G' is flow-well-linked for the terminals, it is possible to route $k/3$ flow units from the terminals in X to the terminals in Y, with congestion at most 2. Since no group U is split between the two sets X and Y, each flow-path must contain at least one edge of $G' \setminus \mathcal{T}$. Therefore, the number of edges in $G' \setminus \mathcal{T}$ is at least $k/6$. $\square$

Families of Good Vertex Subsets. We define a good family of vertex subsets in graph G. We then proceed in two steps. First, we show that we can efficiently find a good family of vertex subsets in graph G. Next, we show that given such good family, we can find the desired routing of the source-sink pairs in $\mathcal{M}$.

Definition 4 *We say that a subset $S \subseteq V(G) \setminus \mathcal{T}$ of vertices is a* good subset *iff there is a subset $\Gamma \subseteq \mathrm{out}_G(S)$ of edges, with $|\Gamma| = k_1$, such that:*

- *S is $\alpha_{\mathrm{WL}}(k)$-well-linked for Γ. That is, for any partition (X, Y) of S, if $\Gamma_X = \Gamma \cap \mathrm{out}(X)$ and $\Gamma_Y = \Gamma \cap \mathrm{out}(Y)$, then $|E_G(X, Y)| \geq \alpha_{\mathrm{WL}}(k) \cdot \min\{|\Gamma_X|, |\Gamma_Y|\}$.*
- *There is a flow F in graph G, where every edge $e \in \Gamma$ sends one flow unit to a distinct terminal $t_e \in \mathcal{T}$ (so for $e \neq e'$, $t_e \neq t_{e'}$), and the congestion caused by F is at most $2\beta(k)/\alpha_{\mathrm{WL}}(k) = O(\log^{4.5} k)$.*

We say that a family $\mathcal{F} = \{S_1, \ldots, S_\gamma\}$ of $\gamma = \gamma_{\mathrm{KRV}}(k) = \Theta(\log^2 k)$ subsets of vertices is good *iff each subset S_j is a good subset of vertices of G, and $S_1, \ldots, S_\gamma$ are pairwise disjoint.*

We view the subset $\Gamma \subseteq \mathrm{out}_G(S)$ of edges as part of the definition of a good subset of vertices. In particular, when we say that we are given a good family $\mathcal{F} = \{S_1, \ldots, S_\gamma\}$ of vertex subsets, we assume that we are also given the corresponding subsets $\Gamma_j \subseteq \mathrm{out}_G(S_j)$ of edges, for all $1 \leq j \leq \gamma$. We use the next theorem, to find a good family of vertex subsets in G.

Theorem 9 *Let G' be a legal contracted graph for G. Then there is an efficient randomized algorithm that w.h.p. either returns a good family $\mathcal{F} = \{S_1, \ldots, S_\gamma\}$ of vertex subsets in G, together with the corresponding subsets $\Gamma_j \subseteq \operatorname{out}_G(S_j)$ of edges for all $1 \leq j \leq \gamma$, or finds a legal contracted graph G'' for G, with $|E(G'')| < |E(G')|$.*

Proof: Let m be the number of edges in $G' \setminus \mathcal{T}$. From Claim 1, $m \geq k/6$. The proof consists of two steps. First, we randomly partition the vertices in $G' \setminus \mathcal{T}$ into γ subsets $X_1, \ldots, X_\gamma$. We show that with high probability, for each $1 \leq j \leq \gamma$, $|\operatorname{out}_{G'}(X_j)| < \frac{10m}{\gamma}$, while the number of edges with both endpoints in X_j, $|E_{G'}(X_j)| \geq \frac{m}{2\gamma^2}$. Therefore, $|E_{G'}(X_j)| > \frac{|\operatorname{out}_{G'}(X_j)|}{20\gamma}$ w.h.p. For each $j : 1 \leq j \leq \gamma$, we then try to recover a good subset S_j of vertices from the cluster X_j. If we succeed, then we obtain a good family $\mathcal{F} = \{S_1, \ldots, S_\gamma\}$ of vertex subsets. If we fail to recover a good vertex subset for some $1 \leq j \leq \gamma$, then we will produce a legal contracted graph G'' containing fewer edges than G'.

We start with the first part. We partition the vertices in $V(G') \setminus \mathcal{T}$ into subsets $X_1, \ldots, X_\gamma$, where each vertex $v \in V(G') \setminus \mathcal{T}$ selects an index $1 \leq j \leq \gamma$ independently uniformly at random, and is then added to X_j. We need the following claim.

Claim 2 *With probability at least $\frac{1}{2}$, for each $1 \leq j \leq \gamma$, $|\operatorname{out}_{G'}(X_j)| < \frac{10m}{\gamma}$, while $|E_{G'}(X_j)| \geq \frac{m}{2\gamma^2}$.*

Proof: Let $H = G' \setminus \mathcal{T}$. Fix some $1 \leq j \leq \gamma$. Let $\mathcal{E}_1(j)$ be the bad event that $\sum_{v \in X_j} d_H(v) \geq \frac{2m}{\gamma} \cdot \left(1 + \frac{1}{\gamma}\right)$. In order to bound the probability of $\mathcal{E}_1(j)$, we define, for each vertex $v \in V(H)$, a random variable x_v, whose value is $\frac{d_H(v)}{k_1}$ if $v \in X_j$ and 0 otherwise. Notice that $x_v \in [0,1]$, and the random variables $\{x_v\}_{v \in V(H)}$ are pairwise independent. Let $B = \sum_{v \in V(H)} x_v$. Then the expectation of B, $\mu_1 = \sum_{v \in V(H)} \frac{d_H(v)}{\gamma k_1} = \frac{2m}{\gamma k_1}$. Using the standard Chernoff bound (see e.g. Theorem 1.1 in [DP09]),

$$\mathbf{Pr}\left[\mathcal{E}_1(j)\right] = \mathbf{Pr}\left[B > (1 + 1/\gamma)\,\mu_1\right] \leq e^{-\mu_1/(3\gamma^2)} = e^{-\frac{2m}{3\gamma^3 k_1}} < \frac{1}{6\gamma}$$

since $m \geq k/6$ and $k_1 = \frac{k}{192\gamma^3 \log \gamma}$.

For each terminal $t \in \mathcal{T}$, let e_t be the unique edge adjacent to t in graph G', and let u_t be its other endpoint. Let $U = \{u_t \mid t \in \mathcal{T}\}$. For each vertex $u \in U$, let $w(u)$ be the number of terminals t, such that $u = u_t$. Notice that $w(u) \leq k_1$ must hold. We say that a bad event $\mathcal{E}_2(j)$ happens iff $\sum_{u \in U \cap X_j} w(u) \geq \frac{k}{\gamma} \cdot \left(1 + \frac{1}{\gamma}\right)$. In order to bound the probability of the event $\mathcal{E}_2(j)$, we define, for each $u \in U$, a random variable y_u, whose value is $w(u)/k_1$ iff $u \in X_j$, and it is 0 otherwise. Notice that $y_u \in [0,1]$, and the variables y_u are independent for all $u \in U$. Let $Y = \sum_{u \in U} y_u$. The expectation of Y is $\mu_2 = \frac{k}{k_1 \gamma}$, and event $\mathcal{E}_2(j)$ holds iff $Y \geq \frac{k}{k_1 \gamma} \cdot \left(1 + \frac{1}{\gamma}\right) \geq \mu_2 \cdot \left(1 + \frac{1}{\gamma}\right)$. Using the standard Chernoff bound again, we get that:

$$\mathbf{Pr}\left[\mathcal{E}_2(j)\right] \leq e^{-\mu_2/(3\gamma^2)} \leq e^{-k/(3k_1\gamma^3)} \leq \frac{1}{6\gamma}$$

since $k_1 = \frac{k}{192\gamma^3 \log \gamma}$. Notice that if events $\mathcal{E}_1(j), \mathcal{E}_2(j)$ do not hold, then:

$$|\operatorname{out}_{G'}(X_j)| \leq \sum_{v \in X_j} d_H(v) + \sum_{u \in U \cap X_j} w(u) \leq \left(1 + \frac{1}{\gamma}\right)\left(\frac{2m}{\gamma} + \frac{k}{\gamma}\right) < \frac{10m}{\gamma}$$

since $m \geq k/6$.

Let $\mathcal{E}_3(j)$ be the bad event that $|E_{G'}(X_j)| < \frac{m}{2\gamma^2}$. We next prove that $\mathbf{Pr}\left[\mathcal{E}_3(j)\right] \leq \frac{1}{6\gamma}$. We say that two edges $e, e' \in E(G' \setminus \mathcal{T})$ are *independent* iff they do not share any endpoints. Our first step is to compute a partition $U_1, \ldots, U_r$ of the set $E(G' \setminus \mathcal{T})$ of edges, where $r \leq 2k_1$, such that for each $1 \leq i \leq r$, $|U_i| \geq \frac{m}{4k_1}$, and all edges in set U_i are mutually independent. In order to compute such a partition, we construct an auxiliary graph Z, whose vertex set is $\{v_e \mid e \in E(H)\}$, and there is an edge $(v_e, v_{e'})$ iff e and e' are not independent. Since the maximum vertex degree in G' is at most k_1, the maximum vertex degree in Z is bounded by $2k_1 - 2$. Using the Hajnal-Szemerédi Theorem [HS70], we can find a partition $V_1, \ldots, V_r$ of the vertices of Z into $r \leq 2k_1$ subsets, where each subset V_i is an independent set, and $|V_i| \geq \frac{|V(Z)|}{r} - 1 \geq \frac{m}{4k_1}$. The partition $V_1, \ldots, V_r$ of the vertices of Z gives the desired partition $U_1, \ldots, U_r$ of the edges of $G' \setminus \mathcal{T}$. For each $1 \leq i \leq r$, we say that the bad event $\mathcal{E}_3^i(j)$ happens iff $|U_i \cap E(X_j)| < \frac{|U_i|}{2\gamma^2}$. Notice that if $\mathcal{E}_3(j)$ happens, then event $\mathcal{E}_3^i(j)$ must happen for some $1 \leq i \leq r$. Fix some $1 \leq i \leq r$. The expectation of $|U_i \cap E(X_j)|$ is $\mu_3 = \frac{|U_i|}{\gamma^2}$. Since all edges in U_i are independent, we can use the standard Chernoff bound to bound the probability of $\mathcal{E}_3^i(j)$, as follows:

$$\mathbf{Pr}\left[\mathcal{E}_3^i(j)\right] = \mathbf{Pr}\left[|U_i \cap E(X_j)| < \mu_3/2\right] \leq e^{-\mu_3/8} = e^{-\frac{|U_i|}{8\gamma^2}}$$

Since $|U_i| \geq \frac{m}{4k_1}$, $m \geq k/6$, $k_1 = \frac{k}{192\gamma^3 \log \gamma}$, and $\gamma = \Theta(\log^2 k)$, this is bounded by $\frac{1}{12k_1\gamma}$. We conclude that $\mathbf{Pr}\left[\mathcal{E}_3^i(j)\right] \leq \frac{1}{12k_1\gamma}$, and by using the union bound over all $1 \leq i \leq r$, $\mathbf{Pr}\left[\mathcal{E}_3(j)\right] \leq \frac{1}{6\gamma}$.

Using the union bound over all $1 \leq j \leq \gamma$, with probability at least $\frac{1}{2}$, none of the events $\mathcal{E}_1(j), \mathcal{E}_2(j), \mathcal{E}_3(j)$ for $1 \leq j \leq \gamma$ happen, and so for each $1 \leq j \leq \gamma$, $|\mathrm{out}_{G'}(X_j)| < \frac{10m}{\gamma}$, and $|E_{G'}(X_j)| \geq \frac{m}{2\gamma^2}$ must hold.
□

Given a partition $X_1, \ldots, X_\gamma$, we can efficiently check whether the conditions of Claim 2 hold. If they do not hold, we repeat the randomized partitioning procedure. From Claim 2, we are guaranteed that w.h.p., after poly(n) iterations, we will obtain a partition with the desired properties. Assume now that we are given the partition $X_1, \ldots, X_\gamma$ of $V(G') \setminus \mathcal{T}$, for which the conditions of Claim 2 hold. Then for each $1 \leq j \leq \gamma$, $|E_{G'}(X_j)| > \frac{|\mathrm{out}_{G'}(X_j)|}{20\gamma}$. Let $X'_j \subseteq V(G) \setminus \mathcal{T}$ be the set obtained from X_j, after we un-contract each cluster, that is, for each super-node $v_C \in V_2 \cap X_j$, we replace v_C with the vertices of C. Notice that $\left\{X'_j\right\}_{j=1}^{\gamma}$ is a partition of $V(G) \setminus \mathcal{T}$. We now proceed as follows. For each $1 \leq j \leq \gamma$, we perform a partitioning procedure for the set X'_j of vertices. We say that this partitioning procedure is successful, iff we find a good subset $S_j \subseteq X'_j$ of vertices. Therefore, if the partitioning procedure is successful for all j, then we obtain a good family $(S_1, \ldots, S_\gamma)$ of disjoint vertex subsets. If the partitioning procedure is not successful for some j, then we will produce a legal contracted graph G'' as required.

We now describe the partitioning procedure for some $j : 1 \leq j \leq \gamma$. Intuitively, we would like to perform a well-linked decomposition of the set X'_j of vertices, using Theorem 6, to obtain a partition $\mathcal{W}_j$ of X'_j. If we could ensure that each set $W \in \mathcal{W}_j$ has $|\mathrm{out}_G(W)| \leq k_1$, and it is $\alpha_{\mathrm{WL}}(k)$-well-linked, then we could simply obtain the graph G'' by first uncontracting all clusters C with $v_C \in V_2 \cap X_j$, and then contracting all clusters in $\mathcal{W}_j$ into super-nodes. Since we are guaranteed that $\sum_{W \in \mathcal{W}_j} |\mathrm{out}_G(W)| \leq |\mathrm{out}_G(X'_j)|(1 + \frac{1}{64\gamma})$, while $|E_{G'}(X_j)| > \frac{|\mathrm{out}_G(X'_j)|}{20\gamma}$, it is easy to verify that $|E(G'')| < |E(G')|$ would hold. There are two problems with this approach. First, in order to use Theorem 6, we need an oracle for finding $(k, \alpha(k))$-violating cuts of sets. Second, even if we had such an oracle, we would not be able to guarantee that for each set $W \in \mathcal{W}$, $|\mathrm{out}_G(W)| \leq k_1$. On the other hand, if, for some set $W \in \mathcal{W}_j$, $|\mathrm{out}_G(W)| \geq k_1$, then it is possible that W is a good set, though this is not guaranteed. Our idea is to gradually perform the well-linked decomposition of the set X'_j, using Theorem 6. We will maintain a partition $\mathcal{W}_j$ of X'_j into clusters, and in addition, a partition of

$\mathcal{W}_j$ into two subsets: $\mathcal{W}^1$ and $\mathcal{W}^2$. Intuitively, $\mathcal{W}^1$ contains all active clusters, that still participate in the well-linked decomposition procedure, and that we may still sub-divide into smaller clusters later, while $\mathcal{W}^2$ contains inactive clusters. In each iteration, we will select an arbitrary cluster $S \in \mathcal{W}^1$, and check if S is a good set of vertices. If so, then we declare the iteration successful, and stop the procedure. Otherwise, we will either obtain a $(k, \alpha(k))$-violating cut of some set $S' \in \mathcal{W}_j$, or we will be able to perform a different well-linked decomposition step that will turn cluster S into an inactive one. We now give a formal description of the partitioning procedure.

Throughout the partitioning procedure, we maintain a partition $\mathcal{W}_j$ of the set X'_j of vertices, where at the beginning $\mathcal{W}_j = \left\{X'_j\right\}$. Set $\mathcal{W}_j$ is in turn partitioned into two subsets: set $\mathcal{W}^1$ of active clusters and set $\mathcal{W}^2$ of inactive clusters. At the beginning, $\mathcal{W}^1 = \mathcal{W}_j$, and $\mathcal{W}^2 = \emptyset$. We also maintain a graph $\tilde{G}$, which is an "almost legal" contracted graph for G in the following sense. The set $V(\tilde{G})$ of vertices is partitioned into two subsets, $\tilde{V}_1 = V(\tilde{G}) \cap V(G)$ and $\tilde{V}_2 = V(\tilde{G}) \setminus \tilde{V}_1$, with $\mathcal{T} \subseteq \tilde{V}_1$. Each vertex $v_C \in \tilde{V}_2$ is associated with a cluster $C \subseteq V(G) \setminus \tilde{V}_1$, and all subsets $\{C\}_{v_C \in \tilde{V}_2}$ of vertices are pairwise disjoint. As before, we can obtain $\tilde{G}$ from G, by contracting every cluster C (where $v_C \in \tilde{V}_2$) into a super-node v_C, and deleting self-loops. For each cluster $S \in \mathcal{W}^1$, there is a super-node $v_S \in \tilde{V}_2$. Let $V'_2 = \left\{v_S \mid S \in \mathcal{W}^1\right\}$ be the set of all such super-nodes. Then for each super-node $v_C \in \tilde{V}_2 \setminus V'_2$, $|\operatorname{out}_G(C)| \leq k_1$, and C is $\alpha_{\text{WL}}(k)$-well-linked for $\operatorname{out}_G(C)$ in graph G. In other words, graph $\tilde{G}$ is a legal contracted graph for G, except for the super-nodes v_S, where $S \in \mathcal{W}^1$: for such nodes v_S, we are not guaranteed that $|\operatorname{out}_G(S)| \leq k_1$, or that S is well-linked. However, if $\mathcal{W}^1 = \emptyset$, then $\tilde{G}$ is a legal contracted graph of G. We remark that for clusters $S \in \mathcal{W}^2$, graph $\tilde{G}$ does not necessarily contain a super-node v_S, and it is possible that the vertices of S are split among several super-nodes. We only maintain the set $\mathcal{W}^2$ for accounting purposes. The initial graph $\tilde{G}$ is obtained from G' as follows: we un-contract all super-nodes $v_C \in X_j$, and then contract all vertices of X'_j into a single super-node $v_{X'_j}$. We set $\mathcal{W}_j = \mathcal{W}^1 = \left\{X'_j\right\}$ and $\mathcal{W}^2 = \emptyset$. While $\mathcal{W}^1$ is non-empty, we select any cluster $S \in \mathcal{W}^1$ and process it. At the end of this procedure, we will either declare that S is a good set, or we will find a $(k, \alpha(k))$-violating cut of some cluster $S' \in \mathcal{W}^1$, or S will become inactive.

Let $S \in \mathcal{W}^1$ be the current cluster. We try to send k_1 flow units from the edges of $\operatorname{out}_{\tilde{G}}(S)$ to the terminals in $\mathcal{T}$ in the current graph $\tilde{G}$ with no congestion. Two case are possible, depending on whether or not such flow exists.

Case 1: Assume first that such flow exists. From the integrality of flow, there is a collection $\mathcal{P}$ of k_1 edge-disjoint paths in $\tilde{G}$, connecting distinct edges in $\operatorname{out}_{\tilde{G}}(S)$ to distinct terminals in $\mathcal{T}$. Let $\Gamma \subseteq \operatorname{out}_{\tilde{G}}(S)$ be the set of k_1 edges which serve as endpoints of the paths in $\mathcal{P}$. We set up an instance of the sparsest cut problem in graph $G[S] \cup \operatorname{out}_G(S)$, where the edges in set Γ serve as terminals. We then run the algorithm $\mathcal{A}_{\text{ARV}}$ on the resulting instance. If the algorithm returns a cut (X, Y) of sparsity less than $\alpha(k)$, then (X, Y) is a $(k, \alpha(k))$-violating cut for S. We then replace S with X and Y in $\mathcal{W}_j$ and in $\mathcal{W}^1$. We also update the current graph $\tilde{G}$, by first un-contracting the super-node v_S, and then contracting the two clusters X and Y into super-nodes v_X and v_Y, respectively. This ends the current iteration, and we then proceed to process some new set in $\mathcal{W}^1$. Assume now that algorithm $\mathcal{A}_{\text{ARV}}$ returns a cut whose sparsity is at least $\alpha(k)$. Then we are guaranteed that S is $\alpha_{\text{WL}}(k) = \alpha(k)/\alpha_{\text{ARV}}(k)$-well-linked for Γ. Recall that we are given a set $\mathcal{P}$ of k_1 edge-disjoint paths connecting the edges in Γ to the terminals $\mathcal{T}$ in graph $\tilde{G}$, where each path connects a distinct edge $e \in \Gamma$ to a distinct terminal $t_e \in \mathcal{T}$. In order for S to be a good set, a low-congestion flow connecting the edges in Γ to the terminals must exist in the original graph G. We will try to find this flow, as follows. The flow will follow the paths in $\mathcal{P}$, except that we need to specify how the flow is routed inside each cluster C for $v_C \in \tilde{V}_2$. Observe that for each such cluster C, the paths in $\mathcal{P}$ define a set

D_C of 1-restricted demands on the edges of $\mathrm{out}_G(C)$. Moreover, the total number of edges in $\mathrm{out}_G(C)$ participating in the paths in $\mathcal{P}$ is at most k_1, as there are only k_1 paths in $\mathcal{P}$ and we can assume w.l.o.g. that they are simple. If $v_C \notin V_2'$, then we are guaranteed that cluster C is $\alpha_{\mathrm{WL}}(k)$-well-linked in graph G. Therefore, we can route the set D_C of demands inside $G[C]$ with congestion at most $2\beta(k)/\alpha_{\mathrm{WL}}(k)$. If $v_C \in V_2'$, then $C \in \mathcal{W}^1$, and it is possible that we cannot route the set D_C of demands inside $G[C]$ with congestion at most $2\beta(k)/\alpha_{\mathrm{WL}}(k)$. We then proceed as follows. If, for each super-node $v_C \in V_2'$, we can route the set D_C of demands inside $G[C]$ with congestion at most $2\beta(k)/\alpha_{\mathrm{WL}}(k)$, then S is a good set, and the jth iteration is successful. Otherwise, let $v_C \in V_2'$ be any super-node, for which such flow does not exist. Consider the instance of the sparsest cut problem defined on the graph $G[C] \cup \mathrm{out}_G(C)$, where the edges of $\mathrm{out}_G(C)$ with non-zero demand serve as terminals (recall that there are at most k_1 such edges). Then the value of the sparsest cut in this instance is at most $\alpha_{\mathrm{WL}}(k)$, and so by applying algorithm $\mathcal{A}_{\mathrm{ARV}}$ on this instance of sparsest cut, we will obtain a $(k, \alpha(k))$-violating cut (X, Y) for set C. We then remove C from $\mathcal{W}^1$ and from $\mathcal{W}_j$, and add X and Y to $\mathcal{W}^1$ and $\mathcal{W}_j$ instead. We also update $\tilde{G}$ by un-contracting the super-node v_C and contracting the clusters X and Y into super-nodes v_X and v_Y, respectively, and end the current iteration. To conclude, if it is possible to send k_1 flow units with no congestion in graph $\tilde{G}$ between $\mathrm{out}_{\tilde{G}}(v_S)$ and $\mathcal{T}$, then either S is a good set, or we find a $(k, \alpha(k))$-violating cut (X, Y) of some cluster $C \in \mathcal{W}^1$ (where possibly $C = S$).

Case 2: Assume now that such flow does not exist. Then there is a cut (X, Y) in graph $\tilde{G}$, where $\mathcal{T} \subseteq Y$, $v_S \in X$, and $|E(X, Y)| < k_1$. (If $|\mathrm{out}_{\tilde{G}}(S)| < k_1$, then we set $X = \{v_S\}$). Let $A \subseteq V(G) \setminus \mathcal{T}$ be the subset of vertices obtained from X after we un-contract every super-node $v_C \in X$. Then $|\mathrm{out}_G(A)| < k_1$. We perform a well-linked decomposition of A, using Corollary 1, and we denote the resulting partition of A by $\mathcal{W}(A)$. Recall that each set $C \in \mathcal{W}(A)$ is guaranteed to be $\alpha_{\mathrm{WL}}(k)$-well-linked, and $|\mathrm{out}_G(C)| < k_1$. Moreover, $\sum_{C \in \mathcal{W}(A)} |\mathrm{out}_G(C)| \le |\mathrm{out}_G(A)| \left(1 + \frac{1}{64\gamma}\right) \le |\mathrm{out}_G(S)| \left(1 + \frac{1}{64\gamma}\right)$. We say that the cluster $S \in \mathcal{W}^1$ is responsible for A, and for the partition $\mathcal{W}(A)$ (we will eventually charge the edges in $\mathrm{out}_G(S)$ for the edges in $\bigcup_{C \in \mathcal{W}(A)} \mathrm{out}_G(C)$). We update the graph $\tilde{G}$, by first un-contracting all super-nodes that belong to X, and then contracting each cluster $C \in \mathcal{W}(A)$ into a super-node v_C. Also, for each vertex $v_C \in \mathcal{W}^1$, if $v_C \in X$, then we move C from $\mathcal{W}^1$ to $\mathcal{W}^2$, where it becomes an inactive cluster (notice that super-node v_C may not exist in the new graph anymore, as the vertices of C may end up being partitioned into several clusters by the contraction procedure). Observe that the cluster S that is responsible for A has been moved from $\mathcal{W}^1$ to $\mathcal{W}^2$ in the current iteration, and hence it becomes an inactive cluster.

This finishes the description of the decomposition procedure for X_j, for $1 \le j \le \gamma$. In order to analyze it, it is enough to show that if this procedure was not declared successful, then the final graph G'', obtained at the end of the procedure, when $\mathcal{W}^1 = \emptyset$, contains fewer edges than G'. (We note that from the above discussion it is clear that G'' must be a legal contracted graph for G.) We bound the number of edges in G'' in two steps. First, we bound the number of edges in $\sum_{C \in \mathcal{W}^2} |\mathrm{out}_G(C)|$. Observe that $\mathcal{W}^2$ defines a partition of the set X_j' of vertices of G. Moreover, this partition was obtained by performing an oracle-based well-linked decomposition of X_j'. Therefore, from Theorem 6, $\sum_{C \in \mathcal{W}^2} |\mathrm{out}_G(C)| \le |\mathrm{out}_G(X_j')| \left(1 + \frac{1}{64\gamma}\right)$.

Next, we bound the number of edges in G'', by charging them to the edges of $\bigcup_{C \in \mathcal{W}^2} \mathrm{out}_G(C)$. Let $A_1, A_2, \ldots, A_\ell$ be all sets of vertices A that were decomposed in iterations where Case 2 happened, in the order in which they were processed. Observe that all vertices of X_j' are contained in $\bigcup_{i=1}^{\ell} A_i$, as all clusters in $\mathcal{W}^2$ are contained in $\bigcup_{i=1}^{\ell} A_i$ (but the sets A_i are not necessarily disjoint). The set of edges of G'' can be partitioned into two subsets: $E_1 = \{e = (u, v) \mid e \in E(G') \cap E(G''); u, v \notin X_j\}$, and set E_2 containing all remaining edges. It is easy to see that $E_2 \subseteq \bigcup_{i=1}^{\ell} (\bigcup_{C \in \mathcal{W}(A_i)} \mathrm{out}_G(C))$. Indeed, let

$e = (u, v) \in E_2$. Let u', v' be the endpoints of the corresponding edge in the original graph G. Two cases are possible. If both $u, v \notin X'_j$, then the only way that edge e was added to the graph $\tilde{G}$ is when either u' or v' belonged to some set A_i. Let i^* be the largest index for which $\{u', v'\} \cap A_{i^*} \neq \emptyset$. Then $e \in \bigcup_{C \in \mathcal{W}(A_{i^*})} \text{out}_G(C)$ must hold. Otherwise, if at least one of the vertices (say v') belongs to X'_j, then, since every vertex in X'_j belongs to some inactive cluster at the end of the algorithm, there is at least one index i such that $v' \in A_i$. Let i^* be the largest index for which $\{u', v'\} \cap A_{i^*} \neq \emptyset$. Then $e \in \bigcup_{C \in \mathcal{W}(A_{i^*})} \text{out}_G(C)$ must hold. Therefore, $E_2 \subseteq \bigcup_{i=1}^{\ell} (\bigcup_{C \in \mathcal{W}(A_i)} \text{out}_G(C))$.

Recall that for each set A_i, for $1 \leq i \leq \ell$, we have a distinct cluster $S_i \in \mathcal{W}^2$ responsible for A_i, and $\sum_{C \in \mathcal{W}(A_i)} |\text{out}_G(C)| \leq |\text{out}_G(S_i)| \left(1 + \frac{1}{64\gamma}\right)$ Therefore, the total number of edges in graph G'' is bounded by:

$$\begin{aligned}
|E(G'')| &\leq |E(G')| - |E_{G'}(X_j)| - |\text{out}_{G'}(X_j)| + |E_2| \\
&\leq |E(G')| - |\text{out}_{G'}(X_j)| \left(1 + \frac{1}{20\gamma}\right) + \sum_{C \in \mathcal{W}_2} |\text{out}_G(C)| \left(1 + \frac{1}{64\gamma}\right) \\
&\leq |E(G')| - |\text{out}_{G'}(X_j)| \left(1 + \frac{1}{20\gamma}\right) + |\text{out}_{G'}(X_j)| \left(1 + \frac{1}{64\gamma}\right)^2 \\
&< |E(G')|
\end{aligned}$$

□

We are now ready to describe the algorithm for finding a good family of vertex subsets in graph G. We start with the graph $G' = G$, which is trivially a legal contracted graph, and repeatedly apply Theorem 9 to it. Since the number of edges in any legal contracted graph is at least $k/6$ by Claim 1, we are guaranteed that after at most $|E(G)|$ iterations, the algorithm will produce a good family of vertex subsets w.h.p. We summarize this algorithm in the following corollary.

Corollary 2 *There is an efficient randomized algorithm that w.h.p. computes a good family of vertex subsets in graph G.*

Finally, we show that given a good family $\mathcal{F}$ of vertex subsets, we can find a routing of $\Omega\left(\frac{k}{\log^{21.5} k \log\log k}\right)$ pairs in $\mathcal{M}$ with congestion at least 14.

We assume that we are given a good family $\mathcal{F} = \{S_1, \ldots, S_\gamma\}$ of vertex subsets of G. For each $1 \leq j \leq \gamma$, we are also given a subset $\Gamma_j \subseteq \text{out}_G(S_j)$ of edges, such that S_j is $\alpha_{\text{WL}}(k)$-well-linked for Γ_j, and there is a flow $F_j : \Gamma_j \leadsto_\eta \mathcal{T}$, where each edge $e \in \Gamma_j$ sends one flow unit to a distinct terminal t_e, and the total congestion due to F_j is at most $\eta = 2\beta(k)/\alpha_{\text{WL}}(k)$.

In order to find the final routing, we build an expander on a subset of terminals and embed it into graph G. More precisely, we select an arbitrary subset $\mathcal{M}' \subseteq \mathcal{M}$ of $k'/2$ source-sink pairs, where $k' = k/\operatorname{poly}\log k$. Let $\mathcal{T}' \subseteq \mathcal{T}$ be the subset of terminals participating in pairs in $\mathcal{M}'$, and assume that $\mathcal{T}' = \{t_1, \ldots, t_{k'}\}$. We construct an expander X on the set $\{v_1, \ldots, v_{k'}\}$ of vertices, which is then embedded into the graph G as follows. For each $1 \leq i \leq k'$, we define a connected component C_i in graph G, that represents the vertex v_i of the expander. For each edge $e = (v_i, v_j) \in E(X)$, we define a path P_e, connecting a vertex of C_i to a vertex of C_j in G. We will ensure that each edge of G may only appear in a small constant number of components C_i, and a small constant number of paths P_e. We also ensure that for each $1 \leq i \leq k'$, terminal $t_i \in C_i$. We will think about the expander vertex v_i as representing the terminal t_i. The idea is that any **vertex-disjoint** routing of the terminal pairs in the expander X can now be translated into a low edge-congestion routing in the original graph G.

We now turn to describe the construction of the expander X and the connected components $C_1, \ldots, C_{k'}$ that we use to embed X into G. The construction exploits the good family $\mathcal{F} = \{S_1, \ldots, S_\gamma\}$ of vertex subsets. For each $1 \le i \le k'$, we construct a collection $T_1, \ldots, T_{k'}$ of trees in graph G. Each such tree T_i contains, for each $1 \le j \le \gamma$, an edge $e_{i,j} \in \Gamma_j$. For each $1 \le j \le \gamma$, the edges $e_{1,j}, e_{2,j}, \ldots, e_{k',j}$ are all distinct, and we think of the edge $e_{i,j}$ as the copy of the vertex $v_i \in V(X)$ for the set S_j. In other words, each tree T_i spans γ copies of the vertex v_i, one copy $e_{i,j}$ for each set $S_j \in \mathcal{F}$. We will ensure that each edge of graph G only participates in a constant number of such trees. Additionally, we build a set $\mathcal{P} = \{P_t \mid t \in \mathcal{T}'\}$ of paths, where path P_t connects the terminal t to a distinct tree T_i (so if $t \neq t'$, then t and t' are connected to different trees), and the total congestion caused by paths in $\mathcal{P}$ is at most 4. We rename the terminals in $\mathcal{T}'$, so that t_i denotes the terminal that is connected to the tree T_i. The final connected component C_i is simply the union of the tree T_i and the path P_{t_i}.

In order to construct the expander X on the set $\{v_1, \ldots, v_{k'}\}$ of vertices, we use the cut-matching game of [KRV06], where we use the sub-graph $G[S_j]$ of G to route the jth matching between the corresponding copies $e_{1,j}, e_{2,j}, \ldots, e_{k',j}$ of the vertices $v_1, \ldots, v_{k'}$, respectively. Recall that we are only guaranteed that sets $\{S_j\}_{j=1}^{\gamma}$ are $\alpha_{\mathrm{WL}}(k)$-well-linked for the edges in Γ_j, and so in order to route these matchings, we may have to incur the congestion of $\Omega(1/\alpha_{\mathrm{WL}}(k))$, which we cannot afford. However, this problem is easy to overcome by performing a suitable grouping of the edges of Γ_j.

The rest of the algorithm proceeds in three steps. In the first step, we perform groupings of the edges in the subsets Γ_j for $1 \le j \le \gamma$. In the second step, we construct the trees $T_1, \ldots, T_{k'}$. In the third step, we finish the construction of the expander X and its embedding into G, and produce the final routing of a subset of demand pairs in $\mathcal{M}'$.

Step 1: Groupings. In this step we compute, for each $1 \le j \le \gamma$, a grouping of the edges in Γ_j. We then establish some properties of these groupings. We use the following two parameters: $p = 8\beta(k)/\alpha_{\mathrm{WL}}(k) = O(\log^{4.5} k)$ is the grouping parameter for the sets Γ_j. The second parameter, $k' = \frac{1}{2\gamma^3} \cdot \lfloor \frac{k_1}{6p} \rfloor = \Omega\left(\frac{k}{\log^{16.5} k \log\log k}\right)$ is the number of the vertices in the expander X that we will eventually construct. We assume w.l.o.g. that k' is an even integer; otherwise we round it down to the closest even integer.

Fix some $1 \le j \le \gamma$. Since $G[S_j] \cup \mathrm{out}_G(S_j)$ is a connected graph, we can find a spanning tree T_j of this graph, and perform a grouping of the edges in Γ_j along this tree into groups whose size is at least p and at most $3p$. Let $\mathcal{G}_j$ be the resulting collection of groups, and let $k^* = \lfloor \frac{k_1}{6p} \rfloor$. For each group $U \in \mathcal{G}_j$, let $T_j(U)$ be the sub-tree of the tree T_j spanning the edges of U. For each group $U \in \mathcal{G}_j$, we select one arbitrary representative edge, and we let Γ'_j denote this set of representative edges. For each $e \in \Gamma'_j$, we denote by U_e the group to which e belongs. Additionally, let $U'_e \subseteq U_e$ be an arbitrary subset of p edges of U_e, including e itself. Notice that $|\Gamma'_j| \ge k^*$ must hold. If $|\Gamma'_j| > k^*$, then we discard edges from Γ'_j arbitrarily, until $|\Gamma'_j| = k^*$ holds. This finishes the description of the grouping. The next theorem establishes some properties of the resulting groupings that will be used later.

Theorem 10

- *For each $1 \le j \le \gamma$, for any pair $X, Y \subseteq \Gamma'_j$ of edge subsets, where $|X| = |Y|$, there is a collection $\mathcal{P}(X,Y) : X \overset{1:1}{\leadsto}_2 Y$ of paths contained in $G[S_j]$, where each path connects a distinct edge of X to a distinct edge of Y, and the paths cause congestion at most 2.*

- *For all $1 \le i, j \le \gamma$, there is a set $\mathcal{P}_{i,j} : \Gamma'_i \overset{1:1}{\leadsto}_2 \Gamma'_j$ of k^* paths in graph G. That is, each path connects a distinct edge of Γ'_i to a distinct edge of Γ'_j, with total congestion at most 2.*

- *Let $\Gamma_1^* \subseteq \Gamma_1'$ be any subset of k' edges, $\mathcal{M}' \subseteq \mathcal{M}$ any subset of $k'/2$ source-sink pairs, and $\mathcal{T}'$ the subset of terminals participating in pairs in $\mathcal{M}'$. Then there is a set $\mathcal{P} : \mathcal{T}' \overset{1:1}{\rightsquigarrow}_4 \Gamma_1^*$ of paths in G, each path connecting a distinct terminal of $\mathcal{T}'$ to a distinct edge of Γ_1^*, with total congestion at most* 4.

Proof: In order to prove the first assertion, fix some $1 \le j \le \gamma$. From the integrality of flow, it is enough to prove that there is a flow $F_j(X,Y)$ in $G[S_j]$, where each edge in X sends one flow unit, each edge in Y receives one flow unit, and the flow congestion is at most 2. We start by defining two subsets $X', Y' \subseteq \Gamma_j$ of edges, as follows: $X' = \bigcup_{e \in X} U_e'$, and $Y' = \bigcup_{e \in Y} U_e'$. Observe that $|X'| = |Y'| = |X| \cdot p$. Since set S_j is $\alpha_{\text{WL}}(k)$-well-linked for Γ_j, there is a flow $F_j(X', Y')$ in $G[S_j]$, where every edge in X' sends one flow unit, every edge in Y' receives one flow unit, and the congestion due to this flow is at most $1/\alpha_{\text{WL}}(k)$. We are now ready to define the flow $F_j(X,Y)$. Each edge $e \in X$ spreads one flow unit uniformly among the edges of U_e' along the tree $T_j(U_e)$. Next, all this flow is sent along the flow-paths in $F_j(X', Y')$, where we scale this flow down by factor p. Finally, each edge $e \in Y$ collects all flow from edges in U_e' along the tree $T_j(U_e)$. Since all trees $\{T_U\}_{U \in \mathcal{G}_j}$ are edge-disjoint, and since the congestion caused by $F_j(X', Y')$ is at most $1/\alpha_{\text{WL}}(k) < p$, the resulting flow $F_j(X,Y)$ causes congestion at most 2.

We now turn to prove the second assertion. From the integrality of flow, it is enough to prove that there is a flow $F_{i,j} : \Gamma_i' \rightsquigarrow_2 \Gamma_j'$, where every edge in Γ_i' sends one flow unit and every edge in Γ_j' receives one flow unit. As before, we construct two edge subsets, $X \subseteq \Gamma_j$ and $Y \subseteq \Gamma_i$, as follows: $X = \bigcup_{e \in \Gamma_i'} U_e'$, and $Y = \bigcup_{e \in \Gamma_j'} U_e'$. Notice that $|X| = |Y| = k^* \cdot p$.

Recall that from the definition of good vertex subsets, we already have a flow F_j, where each edge $e \in \Gamma_j$ sends one flow unit to a distinct terminal in $\mathcal{T}$, with total congestion at most $\eta = 2\beta(k)/\alpha_{\text{WL}}(k)$. We discard all flow-paths except those originating at the edges of X. As a result, we obtain a flow F_j^*, where each edge $e \in X$ sends one flow unit to a distinct terminal $t_e \in \mathcal{T}$, and F_j^* causes congestion at most η in G. Let $\mathcal{T}_j$ be the subset of terminals that receive flow in F_j^*, $|\mathcal{T}_j| = |X|$. Similarly, we can define a flow F_i^*, where each edge $e \in Y$ sends one flow unit to a distinct terminal $t_e \in \mathcal{T}$, and F_i^* causes congestion at most η in G. Subset $\mathcal{T}_i$ of terminals is defined similarly. Notice that $\mathcal{T}_i$ and $\mathcal{T}_j$ are not necessarily disjoint. But since the set $\mathcal{T}$ of terminals is flow-well-linked in G, there is a flow $F : \mathcal{T}_i \overset{1:1}{\rightsquigarrow}_2 \mathcal{T}_j$, where each terminal in $\mathcal{T}_i$ sends one flow unit, each terminal in $\mathcal{T}_j$ receives one flow unit, and the congestion is at most 2. We concatenate the three flows, F_i^*, F, F_j^*, to obtain a flow $F' : X \rightsquigarrow Y$. In this flow, each edge in X sends one flow unit, each edge in Y receives one flow unit, and the total congestion is at most $2\eta + 2$.

We are now ready to define the flow $F_{i,j}$. Each edge $e \in \Gamma_i'$ sends one flow unit along the tree $T_i(U_e)$, which is evenly split among the edges of U_e'. We then use the flow F', scaled down by factor p, to route this flow to the edges of Y. Finally, each edge $e \in \Gamma_j'$ collects the flow that the edges of U_e' receive, along the tree $T_j(U_e)$, so that after collecting all that flow, edge e receives 1 flow unit. In order to analyze the total congestion due to flow $F_{i,j}$, observe that all trees $\{T_i(U)\}_{U \in \mathcal{G}_i} \cup \{T_j(U)\}_{U \in \mathcal{G}_j}$ are edge-disjoint. So the routing along these trees causes a congestion of at most 1. Since flow F' causes congestion of at most $2\eta + 2$, and p is selected so that $p \ge 2\eta + 2$, the congestion due to the scaled-down flow F' is at most 1. The total congestion is therefore at most 2.

Finally, we prove the third assertion. Let $\Gamma_1^* \subseteq \Gamma_1'$ be any subset of k' edges, $\mathcal{M}' \subseteq \mathcal{M}$ any subset of $k'/2$ source-sink pairs, and $\mathcal{T}'$ the set of all terminals participating in the pairs in $\mathcal{M}'$. Let $X = \bigcup_{e \in \Gamma_1^*} U_e'$, so $|X| = k'p$. As before, we make use of the previously defined flow F_1, where each edge $e \in \Gamma_1$ sends one flow unit to a distinct terminal in $\mathcal{T}$, with total congestion at most $\eta = 2\beta(k)/\alpha_{\text{WL}}(k)$. We discard all flow-paths except those that originate at the edges of X. As a result, we obtain a flow F^*, where each edge $e \in X$ sends one flow unit to a distinct terminal $t_e \in \mathcal{T}$, and F^* causes congestion at most

$\eta < p$ in G. We now define a new flow $F^{**} : \Gamma_1^* \rightsquigarrow_2 \mathcal{T}$, where each edge in Γ_1^* sends one flow unit, and each terminal in $\mathcal{T}$ receives at most one flow unit. Flow F^{**} is defined as follows. Each edge $e \in \Gamma_1^*$ sends one flow unit to the edges in set U'_e along the tree $T_1(U_e)$, distributing it evenly among these edges. Each edge in U'_e then sends the $1/p$ flow unit it receives from e to the terminals via the flow F^*, so the flow F^* is scaled down by factor p. Since the congestion caused by flow F^* is $\eta < p$, and the trees $\{T_1(U_e)\}_{e\in\Gamma_1^*}$ are edge-disjoint, the total congestion caused by F^{**} is at most 2. Moreover, each terminal receives at most one flow unit in F^{**}. From the integrality of flow, there is a subset $\mathcal{T}'' \subseteq \mathcal{T}$ of k' terminals, and a collection $\mathcal{P}_1 : \Gamma_1^* \overset{1:1}{\rightsquigarrow}_2 \mathcal{T}''$ of paths in G. Since the set $\mathcal{T}$ of terminals is flow-well-linked, using the integrality of flow, there is a collection $\mathcal{P}_2 : \mathcal{T}'' \overset{1:1}{\rightsquigarrow}_2 \mathcal{T}'$ of paths in G. We then obtain the desired collection $\mathcal{P}$ of paths by concatenating the paths in $\mathcal{P}_1$ with the paths in $\mathcal{P}_2$.$\square$

Step 2: Constructing the Trees. The goal of this step is to find a collection $T_1, \ldots, T_{k'}$ of trees in graph G, such that each edge of G belongs to at most 8 trees. For each tree T_i, we will find a subset $E_i \subseteq E(T_i)$ of *special edges*, that contains, for each $1 \leq j \leq \gamma$, one edge $e_{i,j} \in \Gamma'_j$, such that the sets $E_1, \ldots, E_{k'}$ are pairwise disjoint. Notice that an edge $e \in \Gamma'_j$ may belong to several trees, but only to one of them as a special edge. For each $1 \leq j \leq \gamma$, we denote $\Gamma_j^* = \left\{e_{1,j}, \ldots, e_{k',j}\right\}$, the subset of edges of Γ'_j that the trees $T_1, \ldots, T_{k'}$ contain as special edges. We summarize Step 2 in the next theorem.

Theorem 11 *Given a good family $\mathcal{F}$, and a subset $\Gamma'_j \subseteq \mathrm{out}_G(S_j)$ of edges for each $1 \leq j \leq \gamma$, as computed in Step 1, we can efficiently find k' trees $T_1, \ldots, T_{k'}$ in graph G, and for each tree T_i a subset $E_i \subseteq E(T_i)$ of special edges, such that:*

- *Each edge of G belongs to at most 8 trees;*
- *Subsets $E_1, \ldots, E_{k'}$ of edges are pairwise disjoint; and*
- *For all $1 \leq i \leq k'$, $E_i = \{e_{i,1}, \ldots, e_{i,\gamma}\}$, where for all $1 \leq j \leq \gamma$, $e_{i,j} \in \Gamma'_j$.*

Proof: In order to prove the theorem, we start by augmenting the graph G as follows. First, replace each edge of G with two parallel edges. Next, for each $1 \leq j \leq \gamma$, add a new vertex s_j, and for each edge $e \in \Gamma'_j$, we sub-divide one of the copies of e, by adding a new vertex v_e, which is then connected to the vertex s_j. Notice that from Theorem 10, for each $1 \leq j \neq j' \leq \gamma$, there are exactly k^* edge-disjoint paths connecting s_j to $s_{j'}$ in the resulting graph. Finally, we replace each edge in the resulting graph by two bi-directed edges, thus obtaining a directed Eulerian graph that we denote by G^+. From Theorem 10, for each pair $1 \leq j \neq j' \leq \gamma$ of indices, there are k^* edge-disjoint paths connecting s_j to $s_{j'}$, and k^* edge-disjoint paths connecting $s_{j'}$ to s_j. Notice also that each vertex s_j has exactly k^* incoming edges and exactly k^* outgoing edges.

As a next step, we use the standard edge splitting procedure in graph G^+. Our goal is to eventually obtain a graph $\tilde{H}$ on the set $\{s_1, \ldots, s_\gamma\}$ of vertices, such that each pair $s_j, s_{j'}$ is k^*-edge connected, and each edge $e = (s_j, s_{j'}) \in E(\tilde{H})$ is associated with a path P_e connecting s_j to $s_{j'}$ in G^+, while all paths in $\left\{P_e \mid e \in E(\tilde{H})\right\}$ are edge-disjoint in G^+.

Let $D = (V, A)$ be any directed multigraph with no self-loops. For any pair $(v, v') \in V$ of vertices, their connectivity $\lambda(v, v'; D)$ is the maximum number of edge-disjoint paths connecting v to v' in D. Given a pair $a = (u, v)$, $b = (v, w)$ of edges, a splitting-off procedure replaces the two edges a, b by a single edge (u, w). We denote by $D^{a,b}$ the resulting graph. We use the extension of Mader's theorem [Mad78] to directed graphs, due to Frank [Fra89] and Jackson [Jac98]. Following is a simplified version of Theorem 3 from [Jac98]:

Theorem 12 *Let $D = (V, A)$ be an Eulerian digraph, $v \in V$ and $a = (v, u) \in A$. Then there is an edge $b = (w, v) \in A$, such that for all $y, y' \in V \setminus \{v\}$: $\lambda(y, y'; D) = \lambda(y, y'; D^{ab})$*

We apply Theorem 12 repeatedly to all vertices of G^+ except for the vertices in set $\{s_1, \dots, s_\gamma\}$, until we obtain a directed graph $\tilde{H}$, whose vertex set is $\{s_1, \dots, s_\gamma\}$, and for each $1 \le j, j' \le \gamma$, there are k^* edge-disjoint paths connecting s_j to $s_{j'}$ and k^* edge-disjoint paths connecting $s_{j'}$ to s_j. Clearly, each edge $e = (s_j, s_{j'}) \in E(\tilde{H})$ is associated with a path P_e connecting s_j to $s_{j'}$ in G^+, and all paths $\left\{P_e \mid e \in E(\tilde{H})\right\}$ are edge-disjoint. Let $\tilde{H}'$ denote the undirected multi-graph identical to $\tilde{H}$, except that now all edges become undirected. Notice that each vertex s_j must have $2k^* >> \gamma$ edges adjacent to it in $\tilde{H}'$, so the graph contains many parallel edges. For each pair $s_j, s_{j'}$ of vertices, there are exactly $2k^*$ edge-disjoint paths connecting s_j to $s_{j'}$ in $\tilde{H}'$. For convenience, let us denote $2k^*$ by ℓ.

As a next step, we build an auxiliary undirected graph Z on the set $\{s_1, \dots, s_\gamma\}$ of vertices, as follows. For each pair $s_j, s_{j'}$ of vertices, there is an edge $(s_j, s_{j'})$ in graph Z iff there are at least ℓ/γ^3 edges connecting s_j and $s_{j'}$ in $\tilde{H}'$. If edge $e = (s_j, s_{j'})$ is present in graph Z, then its capacity $c(e)$ is set to be the number of edges connecting s_j to $s_{j'}$ in $\tilde{H}'$. For each vertex s_j, let $C(s_j)$ denote the total capacity of edges incident on s_j in graph Z. We need the following simple observation.

Observation 1

- *For each vertex $v \in V(Z)$, $(1 - 1/\gamma^2)\ell \le C(v) \le \ell$.*
- *For each pair (u, v) of vertices in graph Z, we can send at least $(1 - 1/\gamma)\ell$ flow units from u to v in Z without violating the edge capacities.*

Proof: In order to prove the fist assertion, recall that each vertex in graph $\tilde{H}'$ has ℓ edges incident to it (this is since, in graph G^+, each vertex $s_1, \dots, s_\gamma$ had exactly k^* incoming and k^* outgoing edges, and we did not perform edge splitting on these vertices). So $C(v) \le \ell$ for all $v \in V(Z)$. Call a pair $(s_j, s_{j'})$ of vertices bad iff there are fewer than ℓ/γ^3 edges connecting s_j to $s_{j'}$ in $\tilde{H}'$. Notice that each vertex $v \in V(Z)$ may participate in at most γ bad pairs, as $|V(Z)| = \gamma$. Therefore, $C(v) \ge \ell - \gamma\ell/\gamma^3 = \ell(1 - 1/\gamma^2)$ must hold.

For the second assertion, assume for contradiction that it is not true, and let (u, v) be a violating pair of vertices. Then there is a cut (A, B) in Z, with $u \in A$, $v \in B$, and the total capacity of edges crossing this cut is at most $(1 - 1/\gamma)\ell$. Since u and v were connected by ℓ edge-disjoint paths in graph $\tilde{H}'$, this means that there are at least ℓ/γ edges in graph $\tilde{H}'$ that connect bad pairs of vertices. But since we can only have at most γ^2 bad pairs, and each pair has less than ℓ/γ^3 edges connecting them, this is impossible. □

We now proceed in two steps. First, we show that we can efficiently find a spanning tree of Z with maximum vertex degree at most 3. Next, using this spanning tree, we show how to construct the collection $T_1, \dots, T_{k'}$ of trees.

Claim 3 *There is an efficient algorithm to find a spanning tree T^* of Z with maximum vertex degree at most* 3.

Proof: We use the algorithm of Singh and Lau [SL07] for constructing bounded-degree spanning trees. Suppose we are given a graph $G = (V, E)$, and our goal is to construct a spanning tree T of G, where the degree of every vertex is bounded by B. For each subset $S \subseteq V$ of vertices, let $E(S)$ denote the subset of edges with both endpoints in S, and $\delta(S)$ the subset of edges with exactly one endpoint in S. Singh and Lau consider a natural LP-relaxation for the problem. We note that their algorithm works for a more general problem where edges are associated with costs, and the goal is to find a minimum-cost tree that respects the degree requirements; since we do not need to minimize the tree cost, we only discuss the unweighted version here. For each edge $e \in E$, we have a variable x_e indicating whether e is included in the solution. We are looking for a feasible solution to the following LP.

$$\begin{aligned} \textstyle\sum_{e\in E} x_e &= |V|-1 && && (1)\\ \textstyle\sum_{e\in E(S)} x_e &\le |S|-1 && \forall S\subset V && (2)\\ \textstyle\sum_{e\in\delta(v)} x_e &\le B && \forall v\in V && (3)\\ x_e &\ge 0 && \forall e\in E && (4) \end{aligned}$$

Singh and Lau [SL07] show an efficient algorithm, that, given a feasible solution to the above LP, produces a spanning tree T, where for each vertex $v \in V$, the degree of v is at most $B+1$ in T. Therefore, in order to prove the claim, it is enough to show a feasible solution to the LP, where $B = 2$. Recall that $|V(Z)| = \gamma$. The solution is defined as follows. Let $e = (u, v)$ be any edge in $E(Z)$. We set the LP-value of e to be $x_e = \frac{\gamma-1}{\gamma}\cdot\left(\frac{c(e)}{C(v)}+\frac{c(e)}{C(u)}\right)$. We say that $\frac{\gamma-1}{\gamma}\cdot\frac{c(e)}{C(v)}$ is the contribution of v to x_e, and $\frac{\gamma-1}{\gamma}\cdot\frac{c(e)}{C(u)}$ is the contribution of u. We now verify that all constraints of the LP hold.

First, it is easy to see that $\sum_{e\in E} x_e = \gamma-1$, as required. Next, consider some subset $S \subset V$ of vertices. Notice that it is enough to establish Constraint (2) for subsets S with $|S| \ge 2$. From Observation 1, the total capacity of edges in $E_Z(S, \overline{S})$ must be at least $(1-1/\gamma)\ell$. Since for each $v \in S$, $C(v) \le \ell$, the total contribution of the vertices in S towards the LP-weights of edges in $E_Z(S, \overline{S})$ is at least $\frac{\gamma-1}{\gamma}\cdot(1-1/\gamma) = (1-1/\gamma)^2$. Therefore,

$$\sum_{e\in E(S)} x_e \le \frac{\gamma-1}{\gamma}|S| - (1-1/\gamma)^2 = |S| - |S|/\gamma - 1 - 1/\gamma^2 + 2/\gamma \le |S|-1$$

since we assume that $|S| \ge 2$. This establishes Constraint (2). Finally, we show that for each $v \in V(Z)$, $\sum_{e\in\delta_v} x_e \le 2$. First, the contribution of the vertex v to this summation is bounded by 1. Next, recall that for each $u \in V(Z)$, $C(u) \ge (1-1/\gamma^2)\ell$, while the total capacity of edges in $\delta(v)$ is at most ℓ. Therefore, the total contribution of other vertices to this summation is bounded by $\frac{\ell}{(1-1/\gamma^2)\ell}\cdot\frac{\gamma-1}{\gamma} \le \frac{\gamma}{\gamma+1} \le 1$. The algorithm of Singh and Lau can now be used to obtain a spanning tree T^* for Z with maximum vertex degree at most 3. □

Root the tree T^* at any degree-1 vertex r. Let $e = (s_i, s_j)$ be some edge of the tree, where s_i is the parent of s_j. Recall that there are at least ℓ/γ^3 edges (s_i, s_j) in graph $\tilde{H}'$. Let $A(e)$ be any collection of exactly ℓ/γ^3 such edges. Recall that for each edge $e' \in A(e)$ in graph $\tilde{H}'$, there is a path P, connecting either s_i to s_j or s_j to s_i in graph G^+ (recall that graph G^+ is directed). Since the direction of the edges in G^+ will not play any role in the following argument, we will assume w.l.o.g. that P is directed from s_j towards s_i. Recall that the first edge on path P must connect s_j to some vertex $v_{\tilde{e}}$, where $\tilde{e} \in \Gamma'_j$, and similarly, the last edge on path P connects some vertex $v_{\tilde{e}'}$, for $\tilde{e}' \in \Gamma'_i$ to s_i. So by removing the first and the last edges from path P, we obtain a path $P_{e'}$ in graph G, that connects edge $\tilde{e} \in \Gamma'_j$ to edge $\tilde{e}' \in \Gamma'_i$. Since s_i is the parent of s_j in tree T^*, we will think of $P_{e'}$ as being directed from S_j towards S_i. We call $\tilde{e}$ *the first edge of* $P_{e'}$, and $\tilde{e}'$ *the last edge of* $P_{e'}$. Going back to the edge $e = (s_i, s_j)$ in tree T^*, we can now define a set $\mathcal{P}(e) = \{P_{e'} \mid e' \in A(e)\}$ of exactly ℓ/γ^3 paths in graph G, associated with e. We let

$$B_1(e) = \left\{\tilde{e} \in \Gamma'_j \mid \tilde{e} \text{ is the first edge on some path } P_{e'} \in \mathcal{P}(e)\right\}$$

and

$$B_2(e) = \left\{\tilde{e} \in \Gamma'_i \mid \tilde{e} \text{ is the last edge on some path } P_{e'} \in \mathcal{P}(e)\right\}$$

Both sets $B_1(e)$, $B_2(e)$ are multi-sets, that is, if some edge $\tilde{e} \in \Gamma'_j$ appears as a first edge on two paths in $P_{e'}$, then we add two copies of $\tilde{e}$ to $B_1(e)$. (From the construction of G^+, it is easy to see that $\tilde{e}$ may appear as the first edge on at most two such paths). We then have that $\mathcal{P}(e) : B_1(e) \overset{1:1}{\leadsto}_4 B_2(e)$ in graph G, since, from the construction of graphs G^+ and $\tilde{H}'$, every edge of graph G may appear on at most four paths of $\bigcup_{e \in E(T^*)} \mathcal{P}(e)$.

We call the sets $B_1(e), B_2(e)$ of edges *bundles corresponding to* e, and we view $B_1(e)$ as a bundle that belongs to S_j, while $B_2(e)$ is a bundle that belongs to S_i. Since the degree of tree T^* is at most 3, every set S_j has at most three bundles that belong to it. From the construction of graph G^+, for every vertex $s_i : 1 \leq i \leq \gamma$, each edge in Γ'_i may appear at most twice in the multi-set defined by the union of the bundles that belong to S_i. In particular, it is possible that it appears twice in the same bundle. We need to make sure that this never happens. In order to achieve this, we will define, for each edge $e \in E(T^*)$, smaller bundles, $B'_1(e) \subseteq B_1(e)$ and $B'_2(e) \subseteq B_2(e)$, such that each edge appears at most once in each bundle, and there is a subset $\mathcal{P}'(e) \subseteq \mathcal{P}(e)$, where $\mathcal{P}'(e) : B'_1(e) \overset{1:1}{\leadsto}_4 B'_2(e)$. We will also ensure that $|B'_1(e)| = |B'_2(e)| = \frac{\ell}{4\gamma^3}$.

This is done as follows. Consider some edge $e = (s_i, s_j)$ in tree T^*, and assume that s_i is the parent of s_j in the tree. Consider first $B_1(e)$. For each edge $\tilde{e} \in B_1(e)$, if two copies of $\tilde{e}$ appear in $B_1(e)$, then we remove one of the copies from $B_1(e)$. If $P \in \mathcal{P}(e)$ is one of the two paths for which $\tilde{e}$ is the first edge, then we remove P from $\mathcal{P}(e)$, and we also remove its last edge from $B_2(e)$. It is easy to see that we remove at most half the edges of $B_1(e)$. We then perform the same operation for $B_2(e)$. In the end, both $B_1(e)$ and $B_2(e)$ must contain at least a 1/4 of the original edges, and $\mathcal{P}(e)$ contains at least a 1/4 of the original paths. We now let $\mathcal{P}'(e)$ be any subset of exactly $\ell/4\gamma^3$ remaining paths, and we set $B'_1(e)$ to be the set of all edges $\tilde{e}$ that appear as the first edge on some path in $\mathcal{P}'(e)$, and similarly $B'_2(e)$ the set of all edges that appear as the last edge on some path in $\mathcal{P}'(e)$. We perform this operation for all edges e of tree T^*.

We are now ready to define the subsets $\Gamma^*_j \subseteq \Gamma'_j$ of k' edges, $\Gamma^*_j = \left\{e_{1,j}, \ldots, e_{k',j}\right\}$, that our trees will span. Fix some index $1 \leq j \leq \gamma$. If s_j is not the root of the tree T^*, then we let $\Gamma^*_j = B_1(e)$, where e is the edge connecting s_j to its father in T^*. If s_j is the root of the tree, then $\Gamma^*_j = B_2(e)$, where e is the unique edge incident on s_j in tree T^*. Notice that $|\Gamma^*_j| = \frac{\ell}{4\gamma^3} = \frac{k^*}{2\gamma^3} = k'$.

Finally, we construct the trees $T_1, \ldots, T'_k$. In order to construct these trees, we process the vertices of the tree T^* in the bottom-up order, starting from the leaves. Let s_j be any vertex of T^*, and let $T^*(s_j)$ be the sub-tree of T^*, rooted at s_j. We will ensure that after vertex s_j is processed, we will have a collection $T_1(s_j), \ldots, T_{k'}(s_j)$ of trees, such that for each vertex $s_i \in T^*(s_j)$, each one of the trees contains exactly one distinct edge of Γ^*_i as a special edge. The trees $T_1(s_j), \ldots, T_{k'}(s_j)$ will consist of the union of the paths $\mathcal{P}'(e)$, where e is an edge in the sub-tree $T^*(s_j)$ of T^*, of the edges of G whose both endpoints lie in sets S_i for $s_i \in T^*(s_j)$, and of sets Γ^*_i, for $s_i \in T^*(s_j)$.

Assume first that s_j is a leaf of T^*. Then the trees $T_1(s_j), \ldots, T_k(s_j)$ consist of a single distinct edge of Γ^*_j each. Assume now that s_j is an inner vertex of T^*. We will assume here that s_j has two children, s_a and s_b; the case where s_j only has one child is treated similarly.

Recall that we are given a collection $T_1(s_a), \ldots, T_{k'}(s_a)$ of trees spanning the sets Γ^*_i of vertices s_i in the sub-tree $T^*(s_a)$. We will assume w.l.o.g., that for each such tree $T_q(s_a)$, the root of the tree is an endpoint of the unique edge of Γ^*_a that belongs to $T_q(s_a)$ as a special edge. Let $e = (s_a, s_j)$ be the edge of T^* connecting s_a to s_j. Recall that we are given a collection $\mathcal{P}'(e) : \Gamma^*_a \overset{1:1}{\leadsto} B_2(e)$ of paths in G.

From Theorem 10, we can find a set $\mathcal{P}_1 : B_2(e) \overset{1:1}{\leadsto}_2 \Gamma_j^*$ of paths contained in the sub-graph $G[S_j]$ of G, where each path in $\mathcal{P}_1$ connects a distinct edge of $B_2(e)$ to a distinct edge of Γ_j^*. We now concatenate the paths in $\mathcal{P}'(e)$ with the paths in $\mathcal{P}_1$, to get a collection $\mathcal{P}_1'$ of paths. Each path in $\mathcal{P}_1'$ connects a root of a distinct tree $T_q(s_a)$ to a distinct edge of Γ_j^*.

Similarly, let $e' = (s_b, s_j)$ be the edge of T^* connecting s_b to s_j. We are again given a collection $\mathcal{P}'(e') : \Gamma_b^* \overset{1:1}{\leadsto} B_2(e')$ of paths in G, and we can again find a set $\mathcal{P}_2 : B_2(e') \overset{1:1}{\leadsto}_2 \Gamma_j^*$ of paths contained in $G[S_j]$. Concatenating the paths in $\mathcal{P}'(e')$ and $\mathcal{P}_2$, we again obtain a collection $\mathcal{P}_2'$ of paths, where each path connects a root of a distinct tree $T_q(s_b)$ with a distinct edge in Γ_j^*.

Consider now some edge $\tilde{e} \in \Gamma_j^*$. We have two paths: $P_1 \in \mathcal{P}_1'$, connecting $\tilde{e}$ to the root of some tree $T_q(s_a)$, and path $P_2 \in \mathcal{P}_2'$ connecting $\tilde{e}$ to the root of some tree $T_{q'}(s_b)$. We obtain a tree $T_{\tilde{e}}(s_j)$ by taking the union of $T_q(s_a), T_{q'}(s_b), P_1$ and P_2 (we may need to delete some edges to ensure that it is indeed a tree). The set of the special edges of this new tree consists of all special edges of $T_q(s_a), T_{q'}(s_b)$, and the edge $\tilde{e}$.

At the end of this procedure, when the root r of T^* is processed, we will obtain a desired collection $T_1, \ldots, T_{k'}$ of trees, where for each $1 \le j \le \gamma$, for each $1 \le i \le k'$, tree T_i contains an edge $e_{i,j} \in \Gamma_j^*$, and the edges $e_{1,j}, \ldots, e_{k',j}$ are all distinct. We now analyze the congestion caused by these trees. First, as already observed, each edge of graph G may belong to at most four paths of the set $\bigcup_{e \in E(T^*)} \mathcal{P}'(e)$. Additionally, for each $1 \le j \le \gamma$, we route two subsets of edges of Γ_j' to each other twice. Each such routing causes congestion 2 in graph $G[S_j]$, and so the total congestion caused by all these routings is at most 4. We conclude that each edge of G belongs to at most 8 trees $T_1, \ldots, T_{k'}$.

$\square$

Step 3: Constructing the Expander and finding the routing. In this step, we construct the expander X, together with its embedding into the graph G, and find the final routing of a subset of demands in $\mathcal{M}$. Let $\mathcal{M}' \subseteq \mathcal{M}$ be any subset of $k'/2$ demand pairs, and let $\mathcal{T}'$ be the subset of terminals participating in the pairs of $\mathcal{M}'$.

Let $\mathcal{P} = \mathcal{T}' \overset{1:1}{\leadsto}_4 \Gamma_1^*$ be the collection of paths connecting the terminals of $\mathcal{T}'$ to the edges of $\Gamma_1^* \subseteq \Gamma_1$ (where $\Gamma_1^* = \left\{e_{1,1}, \ldots, e_{k',1}\right\}$), guaranteed by Theorem 10. Denote $\mathcal{P} = \{P_t \mid t \in \mathcal{T}'\}$, where P_t is the path originating from terminal t. Rename the terminals in $\mathcal{T}'$ as $\mathcal{T}' = \{t_1, \ldots, t_{k'}\}$, where for each $1 \le i \le k'$, t_i is the terminal whose path P_t terminates at the edge $e_{i,1}$ (the unique edge of Γ_1^* that belongs to the tree T_i as a special edge). For $1 \le i \le k'$, let C_i be the connected component of graph G, that consists of the union of the tree T_i and the path P_{t_i}. Since each edge of graph G participates in at most 8 trees T_i, and at most 4 paths in $\mathcal{P}$, each edge of G participates in at most 12 connected components C_i.

We now construct the expander X and embed it into the graph G. The set of vertices of X is $V(X) = \{v_1, \ldots, v_{k'}\}$, where we view each vertex v_i as representing the terminal $t_i \in \mathcal{T}'$. We view the connected component C_i as the embedding of the vertex v_i into G. Finally, we need to define the set of the edges of X and specify their embedding into G. In order to do so, we use the cut-matching game of Khandekar, Rao and Vazirani [KRV06] with $\gamma = \gamma_{\text{KRV}}(k)$ iterations. Recall that in each iteration j, the cut player produces a partition (A_j, B_j) of $V(X)$, with $|A_j| = |B_j|$. The matching player then returns some matching M_j between the vertices of A_j and B_j, and the edges of M_j are added to graph X. We are guaranteed that no matter what the matching player does, there is always a way for the cut player to efficiently compute the partitions (A_j, B_j) in each iteration j (which may depend on the previous matchings $M_1, \ldots, M_{j-1}$), such that after γ iterations, X becomes a $\frac{1}{2}$-expander w.h.p. Our idea is to use the graphs $G[S_j]$ to route the matchings M_j. Specifically, let (A_1, B_1) be the partition of $V(X)$ produced by the cut player in the first iteration. Consider the set $\Gamma_1^* = \left\{e_{1,1}, \ldots, e_{k',1}\right\}$ of edges.

Partition (A_1, B_1) of $V(X)$ defines a partition (A'_1, B'_1) of these edges, where $A'_1 = \{e_{i,1} \mid v_i \in A_1\}$ and $B'_1 = \{e_{i,1} \mid v_i \in B_1\}$. From Theorem 10, we can find a set $\mathcal{Q}_1 : A'_1 \overset{1:1}{\rightsquigarrow}_2 B'_1$ of $|A'_1|$ paths contained in $G[S_1]$, where each path in $\mathcal{Q}_1$ connects a distinct edge of A'_1 to a distinct edge of B'_1. Set $\mathcal{Q}_1$ of paths then defines a matching M'_1 between the sets A'_1 and B'_1, which in turn defines a matching M_1 between the sets A_1 and B_1 of vertices of $V(X)$. We then treat M_1 as the response of the matching player. For each edge $e = (v_i, v_{i'}) \in M_1$ of the matching, we let P_e be the unique path of $\mathcal{Q}_1$ connecting $e_{i,1}$ to $e_{i',1}$. We view P_e as the embedding of e into graph G. We continue similarly to execute the remaining iterations, where in each iteration $j : 1 \leq j \leq \gamma$, we use the set $S_j \in \mathcal{F}$ to find the matching M_j. That is, we define the partition (A'_j, B'_j) of Γ'_j based on the partition (A_j, B_j) of $V(X)$ as before, find a collection $\mathcal{Q}_j : A'_j \overset{1:1}{\rightsquigarrow}_2 B'_j$ of paths contained in $G[S_j]$. These paths give us the matching M'_j between the sets A'_j and B'_j of edges, which in turn gives us the matching M_j between the sets A_j and B_j of vertices of $V(X)$. For each edge $e = (v_i, v_{i'}) \in M_j$, we let P_e be the unique path of $\mathcal{Q}_j$ connecting $e_{i,j}$ to $e_{i',j}$. We view P_e as the embedding of e into graph G. The final graph X is the graph obtained after γ iterations, with $E(X) = \bigcup_{j=1}^{\gamma} M_j$, and we are guaranteed that w.h.p. it is a $\frac{1}{2}$-expander. For each edge $e = (v_i, v_{i'}) \in E(X)$, we have defined an embedding P_e of e into G, where P_e is a path connecting some vertex in C_i to some vertex in $C_{i'}$. Let $\mathcal{P}_X = \{P_e \mid e \in E(X)\}$. Then $\mathcal{P}_X = \bigcup_{j=1}^{\gamma} \mathcal{Q}_j$, and the total congestion caused by paths in $\mathcal{P}_X$ in G is at most 2. This finishes the definition of the expander X and of its embedding into G.

We now use the expander X and its embedding into G, to route a subset of demand pairs. We identify from now on the vertices of X with the terminals of $\mathcal{T}'$ they represent, that is, $V(X) = \mathcal{T}'$.

We use Theorem 4 to find a collection $\mathcal{P}$ of $r = \Omega\left(\frac{k'}{\gamma^2 \log k}\right)$ vertex-disjoint paths in the expander X, routing r distinct demand pairs. Let $\mathcal{M}'' \subseteq \mathcal{M}'$ be the set of these demand pairs, and assume w.l.o.g. that $\mathcal{M}'' = \{(t_1, t_2), (t_3, t_4), \ldots, (t_{2r-1}, t_{2r})\}$. For each $1 \leq i \leq r$, let $P_i \in \mathcal{P}$ be the path connecting t_{2i-1} to t_{2i}. In order to complete the routing, we transform each such path P_i into a path Q_i in graph G, connecting the same pair (t_{2i-1}, t_{2i}) of terminals.

Fix some $1 \leq i \leq r$. We now show how to transform the path P_i connecting t_{2i-1} to t_{2i} in graph X to a path Q_i connecting the same pair of terminals in graph G. In order to do so, we will replace the edges and the vertices of path P_i by paths in graph G. First, each edge $e = (t_a, t_b) \in P_i$ is replaced by the path $P_e \subseteq G$, connecting some vertex $v \in C_a$ to some vertex $u \in C_b$. Next, consider some inner vertex $t_x \in P_i$, and let e, e' be the two edges appearing immediately before and immediately after t_x on the original path P_i, respectively. Let $v_x \in C_x$ be the last vertex on path P_e, and let $v'_x \in C_x$ be the first vertex on path $P_{e'}$. Then we replace the vertex t_x with an arbitrary path P_x connecting v_x to v'_x in the connected component C_x of G. It now only remains to take care of the endpoints of path P_i. Let e be the first edge on the original path P_i, and recall that the first vertex on P_i is t_{2i-1}. Let $v_{2i-1} \in C_{2i-1}$ be the first vertex on the path P_e. Then we replace t_{2i-1} by any path connecting t_{2i-1} to v_{2i-1} in the connected component C_{2i-1}. The last vertex of P_i is taken care of similarly. Let Q_i denote the resulting path. Notice that Q_i consists of two types of segments: the first type are the paths P_e for edges $e \in P_i$, and the second type is the paths P_x for vertices $x \in P_i$. Let $Q_1, \ldots, Q_r$ be the resulting set of paths. We now bound the congestion due to paths in $Q_1, \ldots, Q_r$ in graph G. Recall that the paths $\{P_i\}_{i=1}^{r}$ are edge- and vertex-disjoint. Recall also that each edge of graph G participates in at most 2 paths of the set $\mathcal{P}_X = \{P_e \mid e \in E(X)\}$. Therefore, the congestion due to type-1 segments in $\{Q_i\}_{i=1}^{r}$ is at most 2. Since the paths in $\{P_i\}_{i=1}^{r}$ are vertex-disjoint, and every edge of graph G participates in at most 12 components $C_1, \ldots, C_{k'}$, the congestion due to type-2 segments is bounded by 12. Overall, the paths in $\{Q_i\}_{i=1}^{r}$ cause congestion at most 14. The number of demand pairs routed is $r = \Omega\left(\frac{k'}{\gamma^2 \log k}\right) = \Omega\left(\frac{k}{\log^{21.5} k \log\log k}\right)$.

To conclude, we have started with a graph G, a collection $\mathcal{M}$ of k source-sink pairs, and the set $\mathcal{T}$ of terminals participating in pairs in $\mathcal{M}$, such that G is flow-well-linked for $\mathcal{T}$. We have constructed a routing for the subset $\mathcal{M}'' \subseteq \mathcal{M}$ of $\Omega\left(\frac{k}{\log^{21.5} k \log\log k}\right)$ pairs with congestion at most 14.

Since we lose an additional $O(\log^2 k)$ factor on the number of pairs routed due to the pre-processing step that ensures flow-well-linkedness of the terminals, our algorithm routes $\Omega\left(\frac{\mathsf{OPT}}{\log^{23.5} k \log\log k}\right)$ pairs with congestion at most 14 w.h.p.

4 Routing with Grouping

The goal of this section is to prove Theorem 2. We roughly follow the algorithm from Section 3, except that we use a slightly different theorem for routing on expanders, summarized below. Its proof is similar to some arguments from [BFU94], and it uses the new constructive proof of the Lovasz Local Lemma by Moser and Tardos [MT10]. The proof is deferred to Section D of the Appendix.

Theorem 13 *Let $G = (V, E)$ be any n-vertex α-expander (for $\alpha \le 1$) with maximum degree $d_{\max}$, and let $c \ge 1$ be any integer. Then there is a value $m = \Theta\left(d_{\max}^{1+3/c}(\log n)^{1+5/c}/\alpha^{1+3/c}\right)$, such that, for any partition $\mathcal{G} = (V_1, \ldots, V_r)$ of the vertices of G into groups of size at least m, and for any partial matching $M \subseteq ([r] \times [r])$, we can efficiently find, for each pair $(i, j) \in M$, a path $P_{i,j}$ connecting a vertex of V_i to a vertex of V_j, such that w.h.p., the set of paths $\{P_{i,j} \mid (i, j) \in M\}$ causes* **vertex congestion** *at most c in G.*

Assume that we are given a graph $G = (V, E)$ and a set $\mathcal{T} \subseteq V$ of k_0 terminals, such that G is α_0-well-linked for $\mathcal{T}$. We will construct an expander X on a subset of terminals in $\mathcal{T}$ as in Section 3, where X is a $\frac{1}{2}$-expander, $|V(X)| \le k_0$, and the maximum degree of X is bounded by $\gamma_{\text{KRV}}(k_0) = O(\log^2 k_0)$. We denote by $m = \Theta\left((\log k_0)^{3+11/c}\right)$ the corresponding parameter from Theorem 4 for this setting.

We also use the following parameters. Let $\tilde{k} = \frac{k_0\alpha_0}{12\beta_{\text{FCG}}(k_0)} = \Omega\left(\frac{k_0\alpha_0}{\log k_0}\right)$. Recall that we have defined, in Section 3, a parameter k', whose value is $\Omega\left(\frac{k}{\log^{16.5} k \log\log k}\right)$, where k is the number of the terminals. We define a function $q(k) = O(\log^{16.5} k \log\log k)$, so that for any integer k, $k' = k/q(k)$. We then set $\tilde{k}' = \tilde{k}/q(\tilde{k}) = \Omega\left(\frac{k_0\alpha_0}{\log k_0 \cdot \log^{16.5} \tilde{k} \log\log \tilde{k}}\right) = \Omega\left(\frac{k_0\alpha_0}{\log^{18} k_0}\right)$. Intuitively, we will define a grouping $\mathcal{G}''$ of the terminals in $\mathcal{T}$ into groups of size roughly $k_0\alpha_0/\beta_{\text{FCG}}(k_0)$, and select one representative terminal from each group. Let $\mathcal{T}''$ denote the resulting set of terminals. We will show that the set $\mathcal{T}''$ is flow-well-linked in graph G, and $|\mathcal{T}''| \ge \tilde{k}$. We can then apply the algorithm from Section 3 to construct an expander X on a subset $\mathcal{T}' \subseteq \mathcal{T}''$ of $\tilde{k}'$ terminals. These terminals are in turn grouped into groups of size at least m, and we then apply Theorem 4 to route these terminals in the expander X. We now proceed with a formal description of the algorithm.

We define three hierarchical groupings of the terminals in $\mathcal{T}$. Let T be any spanning tree of the graph G. Our first step is to group the terminals in $\mathcal{T}$ into groups of size roughly $k_0 m/\tilde{k}'$. To do so, we use the grouping technique with the parameter $6\lceil k_0 m/\tilde{k}'\rceil$ on the set $\mathcal{T}$ of terminals and the tree T. As a result, we obtain a partition $\mathcal{G}$ of the set $\mathcal{T}$ of terminals into groups of size at least $6\lceil k_0 m/\tilde{k}'\rceil$, and at most $18\lceil k_0 m/\tilde{k}'\rceil$. For each group $U \in \mathcal{G}$, there is a tree T_U spanning the terminals of U, and the all trees in $\{T_U\}_{U\in\mathcal{G}}$ are edge-disjoint. The final grouping of the terminals returned by the algorithm is $\mathcal{G}$. The size of each group in $\mathcal{G}$ is bounded by $18\lceil k_0 m/\tilde{k}'\rceil = O\left(k_0(\log k_0)^{3+11/c} \cdot \frac{\log^{18} k_0}{k_0\alpha_0}\right) = O\left(\frac{(\log k_0)^{21+11/c}}{\alpha_0}\right)$, as required. Assume now that we are given a set $\mathcal{M}$ of integral $(1, \mathcal{G})$-restricted demands on $\mathcal{T}$. We now show an algorithm to integrally route the demands in $\mathcal{M}$.

For each group $U \in \mathcal{G}$, we further partition the terminals in U into at least m groups of roughly equal size, using the tree T_U. Let $n_U = |U|$. We use the grouping technique with the parameter $\frac{1}{3}\lfloor n_U/m \rfloor$ for U and tree T_U. We then obtain at least m groups, whose sizes are at least $\frac{k_0}{\tilde{k}'}$ and at most $\frac{36k_0}{\tilde{k}'}$. For each group $U \in \mathcal{G}$, let $\mathcal{P}(U)$ denote the resulting partition of U, and let $\mathcal{G}' = \bigcup_{U \in \mathcal{G}} \mathcal{P}(U)$ be the corresponding grouping of the terminals. Notice that again for each group $U' \in \mathcal{G}'$, we have a tree $T_{U'}$ spanning the terminals of U', such that all trees in $\{T_{U'}\}_{U' \in \mathcal{G}'}$ are edge-disjoint.

Finally, for each group $U' \in \mathcal{G}'$, we further partition the terminals in U' into groups of size at least $\lceil \frac{\beta_{\mathrm{FCG}}(k_0)}{\alpha_0} \rceil$ and at most $3\lceil \frac{\beta_{\mathrm{FCG}}(k_0)}{\alpha_0} \rceil$, using the standard grouping technique on the tree $T_{U'}$, with the parameter $\lceil \frac{\beta_{\mathrm{FCG}}(k_0)}{\alpha_0} \rceil$. For each set $U' \in \mathcal{G}'$, let $\mathcal{P}'(U')$ be the resulting partition of U', and let $\mathcal{G}'' = \bigcup_{U' \in \mathcal{G}'} \mathcal{P}'(U')$ be the resulting partition of the terminals. For each set $U'' \in \mathcal{G}''$, let $t_{U''}$ be any representative terminal from U'', and let $\mathcal{T}'' = \{t_{U''} \mid U'' \in \mathcal{G}''\}$. Each group $U'' \in \mathcal{G}''$ is again associated with a tree $T_{U''}$ spanning the terminals of U'', and all trees in $\{T_{U''}\}_{U'' \in \mathcal{G}''}$ are edge-disjoint. We start with the following simple claim.

Claim 4 *The terminals in $\mathcal{T}''$ are flow-well-linked.*

Proof: Let M' be any partial matching on the terminals of $\mathcal{T}''$. We extend the matching M' as follows. Assume w.l.o.g. that $M' = \{(t_1, t_2), \ldots, (t_{2r-1}, t_{2r})\}$, and for all $1 \le i \le 2r$, $t_i \in U_i$, where $U_i \in \mathcal{G}''$. For each $1 \le j \le r$, let M_j be any matching of size $\lceil \frac{\beta_{\mathrm{FCG}}(k_0)}{\alpha_0} \rceil$ between U_{2j-1} and U_{2j}, and let $M'' = \bigcup_{j=1}^{r} M_j$. Since graph G is α_0-well-linked for $\mathcal{T}$, matching M'' can be fractionally routed with congestion at most $\beta_{\mathrm{FCG}}(k_0)/\alpha_0$ in G. Let F be the resulting flow, scaled down by factor $\lceil \frac{\beta_{\mathrm{FCG}}(k_0)}{\alpha_0} \rceil$. Then F causes congestion at most 1, and for each $1 \le j \le r$, the terminals in U_{2j-1} send one flow unit to the terminals in U_{2j}.

For each pair $(t_{2j-1}, t_{2j}) \in M'$, we define the flow from t_{2j-1} to t_{2j} as follows: terminal t_{2j-1} spreads one unit of flow along the tree $T_{U_{2j-1}}$ to the terminals in U_{2j-1}, where the amount of flow each terminal receives equals to the amount of flow it sends in F. This flow is then concatenated with the flow originating from the terminals in U_{2j-1} in F, and finally t_{2j} collects one flow unit from the terminals in U_{2j} via the tree $T_{U_{2j}}$. It is easy to see that the resulting flow causes congestion at most 2. □

Notice that the number of terminals in $\mathcal{T}''$ is $|\mathcal{T}''| \ge \frac{k_0}{3\lceil \beta_{\mathrm{FCG}}(k_0)/\alpha_0 \rceil} \ge \frac{k_0 \alpha_0}{12 \beta_{\mathrm{FCG}}(k_0)} \ge \tilde{k}$. Therefore, we now have a graph G and a subset $\mathcal{T}''$ of at least $\tilde{k}$ terminals, that are flow-well-linked in G. This is precisely the starting point of the algorithm in Section 3. We can now use the algorithm from Section 3 to construct the expander X on a subset $\mathcal{T}' \subseteq \mathcal{T}''$ of $\tilde{k}'$ terminals. The only difference is that, instead of selecting an arbitrary subset $\mathcal{T}'$ of terminals as in the algorithm, we select $\mathcal{T}'$ as follows. Consider the grouping $\mathcal{G}'$ of the terminals. Let $\mathcal{G}^*$ be the grouping of the terminals in $\mathcal{T}''$ that $\mathcal{G}'$ induces. We select one representative terminal from each group in $\mathcal{G}^*$, and we let $\mathcal{T}'$ be the set of all selected terminals. By our construction, $|\mathcal{T}'| = |\mathcal{G}'| \le 6m|\mathcal{G}| \le \frac{6k_0 m}{6k_0 m/\tilde{k}'} \le \tilde{k}'$.

We now use the algorithm from Section 3 to construct a $\frac{1}{2}$-expander X on the set $\mathcal{T}'$ of terminals, and embed it into G. Recall that for each terminal $t \in \mathcal{T}'$, we have a connected component C_t of G, and each edge of G participates in at most 12 such components. Each edge e of X is mapped to a path P_e in G, and each edge of G participates in at most two such paths.

Consider the grouping $\mathcal{G}^{**}$ of the terminals in $\mathcal{T}'$, induced by $\mathcal{G}$. In order to obtain $\mathcal{G}^{**}$, we start from $\mathcal{G}$, and we ignore terminals that do not belong to $\mathcal{T}'$. By our construction, each group in $\mathcal{G}^{**}$ contains at least m terminals from $\mathcal{T}'$. We assume w.l.o.g. that the input set of demands is $\mathcal{M} = \{(t_1, t_2), \ldots, (t_{2r-1}, t_{2r})\}$, and for each $1 \le i \le 2r$, $t_i \in U_i$, where $U_i \in \mathcal{G}$. Let $U_i^* = U_i \cap \mathcal{T}'$, and recall that $|U_i^*| \ge m$, and $U_i^* \in \mathcal{G}^{**}$. We now use Theorem 4 on graph X, set $\mathcal{T}'$ of terminals, grouping

$\mathcal{G}^{**}$, and matching $M' = \{(1,2),(3,4),\ldots,(2r-1,2r)\}$. Let $\mathcal{P}''$ be the set of paths returned by the theorem, where for each $1 \leq j \leq r$, there is a path $P''_j \in \mathcal{P}$ connecting some terminal $t'_{2j-1} \in U_{2j-1}$ to some terminal $t'_{2j} \in U_{2j}$. The paths in $\mathcal{P}''$ cause vertex congestion at most c in X. We transform these paths into a set $\mathcal{P}'$ of paths connecting the same pairs of terminals in graph G. Since each edge of G participates in at most 12 connected component, and at most two paths in set $\{P_e \mid e \in E(X)\}$, the total congestion caused by paths in $\mathcal{P}'$ is at most $14c$. For each $1 \leq j \leq r$, let $P'_j \in \mathcal{P}'$ be the path connecting t'_{2j-1} to t'_{2j}.

We then construct a path P_j connecting t_{2j-1} to t_{2j} as follows: first connect t_{2j-1} to t'_{2j-1} via the tree $T_{U_{2j-1}}$, then use the path P'_j to connect t'_{2j-1} to t'_{2j}, and finally connect t'_{2j} to t_{2j} via the tree $T_{U_{2j}}$. Let $\mathcal{P} = \{P_j\}_{j=1}^r$ be the final routing. Then the total edge congestion caused by $\mathcal{P}$ is bounded by $14c + 1$.

5 Integral Sparsifiers

In this section we prove Theorem 3. Notice that we can assume w.l.o.g. that the degree of every terminal is 1, and the number of terminals is d: for each terminal $t \in \mathcal{T}$, we can simply sub-divide every edge e incident on t with a new vertex v_t, let S_t be the set of these new vertices, and set $\mathcal{T}' = \bigcup_{t\in\mathcal{T}} S_t$. Let G' be the sub-graph of the resulting graph induced by $(V \setminus \mathcal{T}) \cup \mathcal{T}'$, and let $\mathcal{T}'$ be the new set of terminals. Then every terminal in $\mathcal{T}'$ has degree 1, and $|\mathcal{T}'| = d$. Moreover, if H' is a quality (q_1, q_2) integral sparsifier for G', we can obtain a sparsifier H for G by unifying, for each $t \in \mathcal{T}$, all vertices in S_t into a single vertex t in graph H'. It is immediate to verify that the resulting graph H is a quality (q_1, q_2)-sparsifier for G.

For convenience, from now on we assume that every terminal in $\mathcal{T}$ has degree 1, and we denote by k the number of terminals in $\mathcal{T}$. We now show a construction of a sparsifier for G of size $O(k)$.

We first consider a special case where graph G is $\alpha_{\text{WL}}(k)$-well-linked for the set $\mathcal{T}$ of terminals. We use Theorem 2 with $c = 1$ and $\alpha_0 = \alpha_{\text{WL}}(k)$ to find a partition $\mathcal{G}$ of the terminals into subsets of size $Z = O(\log^{32} k/\alpha_{\text{WL}}(k)) = O(\log^{35.5} k)$. Recall that in the proof of Theorem 2, we have constructed, for each group $U \in \mathcal{G}$, a tree $T_U \subseteq G$ containing all terminals of U, such that the trees $\{T_U\}_{U\in\mathcal{G}}$ are edge-disjoint.

Consider some group $U \in \mathcal{G}$ and its corresponding tree T_U. We construct a new tree T'_U, which is a minor of T_U, as follows. Root T_U at some arbitrary vertex r_U. For each vertex v of T_U, let $T(v)$ be the sub-tree of T_U rooted at v, and let $S(v)$ be the set of vertices of T_U, excluding v. While T_U contains a vertex v with $S(v) \cap U = \emptyset$, we delete all vertices of $S(v)$ from the tree T_U. Assume now that for each vertex v of T_U, $S(v)$ contains some vertex of U. While T_U contains any degree-2 vertex $v' \neq r_U$, we replace the two edges incident on v' with a single edge. Let T'_U be the resulting tree. It is easy to see that T'_U is a minor of T_U, and it contains at most $2|U|$ vertices, since its leaves belong to U. In order to construct the sparsifier H, we start with disjoint copies of trees T'_U (so if any vertex is contained in several such trees, we use several copies of this vertex). Finally, we add a new vertex r, and an edge (r, r_U) for every $U \in \mathcal{G}$, connecting r to the root of the tree T'_U. We claim that graph H is a quality-$(Z, 31)$ integral flow sparsifier for G.

Indeed, let D be any set of demands on $\mathcal{T}$. By scaling D appropriately, we can assume w.l.o.g. $\eta(G, D) = 1$. Let D' be the demand set obtained from D by scaling all demands down by factor Z. We show that $\eta(H, D') \leq 1$. For each group $U \in \mathcal{G}$, the total demand originating from the terminals of U is at most 1. For each pair $t, t' \in U$, we route the demand $D'(t, t')$ along the tree T'_U. For each pair (t, t') with $t \in U$, $t' \in U'$, where $U \neq U'$, we route $D'(t, t')$ flow units from t to r_U along the tree T'_U, then use the edges (r_U, r) and $(r, r_{U'})$, and finally we route $D'(t, t')$ flow units from $r_{U'}$ to t' along the tree $T'_{U'}$. This gives a routing of D' with congestion at most 1. Therefore, $\eta(H, D) \le Z$.

Assume now that we are given some collection $\mathcal{M}$ of pairs of terminals, and a set $\mathcal{P}$ of paths that connects the pairs of terminals in $\mathcal{M}$ with congestion at most η in graph H. We show a collection $\mathcal{P}'$ of paths connecting the same pairs of terminals with congestion at most 31η in graph G.

We decompose $\mathcal{M}$ into two subsets: $\mathcal{M}_1 \subseteq \mathcal{M}$ containing pairs (s,t) where both s and t belong to the same group U, and $\mathcal{M}_2 = \mathcal{M} \setminus \mathcal{M}_1$. For each group $U \in \mathcal{G}$, let $\mathcal{M}_1(U) \subseteq \mathcal{M}_1$ be the set of pairs that belong to group U. Then we can assume w.l.o.g. that all pairs in $\mathcal{M}_1(U)$ are routed along the tree T'_U in H, and the total congestion of this routing is at most η. We can then route these pairs along the tree T_U in graph G. We therefore obtain an integral routing of all pairs in $\mathcal{M}_1$ with congestion at most η in graph G.

We now turn to pairs in $\mathcal{M}_2$. Since the pairs in $\mathcal{M}_2$ can be routed in graph H with congestion at most η, for each $U \in \mathcal{G}$, the terminals of U participate in at most η pairs in $\mathcal{M}_2$. We decompose $\mathcal{M}_2$ into 2η subsets $\mathcal{M}^1, \ldots, \mathcal{M}^{2\eta}$, such that for each $1 \le j \le 2\eta$, for each $U \in \mathcal{G}$, at most one terminal of U participates in pairs in $\mathcal{M}^j$. Such a decomposition can be found greedily. We consider each pair $(s,t) \in \mathcal{M}_2$ in turn. Assume that $s \in U$, $t \in U'$. We select any index j, such that no terminal of $U \cup U'$ participates in any pair of $\mathcal{M}^j$, and add (s,t) to $\mathcal{M}^j$. Since for each $U \in \mathcal{G}$, the terminals of U participate in at most η pairs in $\mathcal{M}_2$, it is easy to see that this greedy process will give the desired decomposition. For each $1 \le j \le 2\eta$, the pairs in $\mathcal{M}^j$ now define a set D_j of $(1, \mathcal{G})$-restricted demands. Using Theorem 2, there is an efficient algorithm that w.h.p. finds a routing of D_j in G with congestion at most 15. Therefore, we obtain a routing of all pairs in $\mathcal{M}_2$ with congestion at most 30η. Overall, we route all pairs in $\mathcal{M}$ with congestion at most 31η. This concludes the proof that H is a quality-$(Z, 31)$ integral flow sparsifier for G. Notice that $|V(H)| \le 2k$.

We now consider a general case, where we are given a graph $G = (V, E)$, and set $\mathcal{T}$ of k terminals, such that the degree of every terminal is 1 in G, but G is not necessarily well-linked for $\mathcal{T}$. We compute a well-linked decomposition $\mathcal{W}$ of $V(G) \setminus \mathcal{T}$ using Corollary 1. Let G' be the graph obtained from G by subdividing every edge $e \in \bigcup_{W \in \mathcal{W}} \operatorname{out}(W)$ by a vertex v_e. For each cluster $W \in \mathcal{W}$, let $\mathcal{T}_W = \{v_e \mid e \in \operatorname{out}_G(W)\}$, and let $G_W = G'[W \cup \mathcal{T}_W]$. Notice that since W is $\alpha_{\text{WL}}(k)$-well-linked, we are guaranteed that graph G_W is $\alpha_{\text{WL}}(k)$-well-linked for the set $\mathcal{T}_W$ of terminals. We can then compute a sparsifier H_W for G_W as before.

In order to obtain the final sparsifier H, we replace, for each $W \in \mathcal{W}$, graph G_W with graph H_W in G'. In order to do so, we delete all vertices of W from G', and add the vertices and the edges of H_W to it. Finally, for each $t \in \mathcal{T}_W$, we identify the two copies of t in the resulting graph. It is immediate to verify that the resulting graph H is a quality-$(Z, 31)$ integral flow sparsifier for $(G, \mathcal{T})$, using the fact that for each $W \in \mathcal{W}$, graph H_W is a quality-$(Z, 31)$ integral flow sparsifier for $(G_W, \mathcal{T}_W)$.

Acknowledgements The author thanks Matthew Andrews and Sanjeev Khanna for many inspiring discussions about routing problems.

References

[ACG+10] 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.

[And10] Matthew Andrews. Approximation algorithms for the edge-disjoint paths problem via Raecke decompositions. In *Proceedings of the 2010 IEEE 51st Annual Symposium on*

Foundations of Computer Science, FOCS '10, pages 277–286, Washington, DC, USA, 2010. IEEE Computer Society.

[AR98] Yonatan Aumann and Yuval Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. Comput.*, 27(1):291–301, 1998.

[AR01] Yossi Azar and Oded Regev. Strongly polynomial algorithms for the unsplittable flow problem. In *In Proceedings of the 8th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 15–29, 2001.

[ARV09] Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2), 2009.

[AZ05] Matthew Andrews and Lisa Zhang. Hardness of the undirected edge-disjoint paths problem. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 276–283. ACM, 2005.

[AZ07] Matthew Andrews and Lisa Zhang. Hardness of the undirected congestion minimization problem. *SIAM J. Comput.*, 37(1):112–131, 2007.

[AZ08] Matthew Andrews and Lisa Zhang. Almost-tight hardness of directed congestion minimization. *J. ACM*, 55(6), 2008.

[BFSU94] Andrei Z. Broder, Alan M. Frieze, Stephen Suen, and Eli Upfal. Optimal construction of edge-disjoint paths in random graphs. In *Proc. 5th ACM-SIAM SODA*, pages 603–612, 1994.

[BFU94] Andrei Z. Broder, Alan M. Frieze, and Eli Upfal. Existence and construction of edge-disjoint paths on expander graphs. *SIAM J. Comput.*, pages 976–989, 1994.

[BS00] Alok Baveja and Aravind Srinivasan. Approximation algorithms for disjoint paths and related routing and packing problems. *Mathematics of Operations Research*, 25:2000, 2000.

[CGKT07] Julia Chuzhoy, Venkatesan Guruswami, Sanjeev Khanna, and Kunal Talwar. Hardness of routing with congestion in directed graphs. In *ACM Symposium on Theory of Computing*, pages 165–178, 2007.

[CK03] Chandra Chekuri and Sanjeev Khanna. Edge disjoint paths revisited. In *Symposium on Discrete Algorithms*, pages 628–637, 2003.

[CKS04] Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. The all-or-nothing multicommodity flow problem. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, STOC '04, pages 156–165, New York, NY, USA, 2004. ACM.

[CKS05] Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. Multicommodity flow, well-linked terminals, and routing problems. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 183–192, New York, NY, USA, 2005. ACM.

[CKS06a] Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. Edge-disjoint paths in planar graphs with constant congestion. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, STOC '06, pages 757–766, New York, NY, USA, 2006. ACM.

[CKS06b] 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.

[CLLM10] Moses Charikar, Tom Leighton, Shi Li, and Ankur Moitra. Vertex sparsifiers and abstract rounding algorithms. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 265–274, Washington, DC, USA, 2010. IEEE Computer Society.

[CMS07] Chandra Chekuri, Marcelo Mydlarz, and F. Bruce Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Trans. Algorithms*, 3, August 2007.

[DP09] Devdatt Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[EGK+10] Matthias Englert, Anupam Gupta, Robert Krauthgamer, Harald Räcke, Inbal Talgam-Cohen, and Kunal Talwar. Vertex sparsifiers: new results from old techniques. In *Proceedings of the 13th international conference on Approximation, and 14 the International conference on Randomization, and combinatorial optimization: algorithms and techniques*, APPROX/RANDOM'10, pages 152–165, Berlin, Heidelberg, 2010. Springer-Verlag.

[Fra85] A. Frank. Edge-disjoint paths in planar graphs. *Journal of Combinatorial Theory*, 39:164–178, 1985.

[Fra89] A. Frank. On connectivity properties of Eulerian digraphs. *Ann. Discrete Math.*, 41, 1989.

[Fri00] Alan M. Frieze. Edge-disjoint paths in expander graphs. *SIAM Journal On Computing*, 30:2001, 2000.

[GKR+99] Venkatesan Guruswami, Sanjeev Khanna, Rajmohan Rajaraman, Bruce Shepherd, and Mihalis Yannakakis. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. In *Journal of Computer and System Sciences*, page pages, 1999.

[GVY93] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees, with applications to matching and set cover. In Andrzej Lingas, Rolf G. Karlsson, and Svante Carlsson, editors, *ICALP*, volume 700 of *Lecture Notes in Computer Science*, pages 64–75. Springer, 1993.

[GVY95] N. Garg, V.V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)-cut theorems and their applications. *SIAM Journal on Computing*, 25:235–251, 1995.

[HS70] A. Hajnal and E. Szemerédi. Proof of a conjecture of P. Erdos. *Combinatorial Theory and its Applications*, pages 601–623, 1970.

[Jac98] Bill Jackson. Some remarks on arc-connectivity, vertex splitting, and orientation in graphs and digraphs. *J. Graph Theory*, 12:429–436, 1998.

[Kar72] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[KK10] Ken-ichi Kawarabayashi and Yusuke Kobayashi. An $O(\log n)$-approximation algorithm for the disjoint paths problem in Eulerian planar graphs and 4-edge-connected planar graphs. In *APPROX-RANDOM'10*, pages 274–286, 2010.

[KK11] Ken-ichi Kawarabayashi and Yusuke Kobayashi. Breaking $O(n^{1/2})$-approximation algorithms for the edge-disjoint paths problem with congestion two. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*, pages 81–88. ACM, 2011.

[Kle96] Jon Kleinberg. Approximation algorithms for disjoint paths problems, 1996.

[Kle05] Jon M. Kleinberg. An approximation algorithm for the disjoint paths problem in even-degree planar graphs. In *FOCS'05*, pages 627–636, 2005.

[KR96] Jon Kleinberg and Ronitt Rubinfeld. Short paths in expander graphs. In *In Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 86–95, 1996.

[KRV06] Rohit Khandekar, Satish Rao, and Umesh V. Vazirani. Graph partitioning using single commodity flows. In Jon M. Kleinberg, editor, *STOC*, pages 385–390. ACM, 2006.

[KS04] Stavros G. Kolliopoulos and Clifford Stein. Approximating disjoint-path problems using packing integer programs. *Mathematical Programming*, 99:63–87, 2004.

[KT95] Jon Kleinberg and Éva Tardos. Approximations for the disjoint paths problem in high-diameter planar networks. In *ACM Symposium on Theory of Computing*, pages 26–35, 1995.

[LLR94] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Proceedings of 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 577–591, 1994.

[LM10] F. Thomson Leighton and Ankur Moitra. Extensions and limits to vertex sparsification. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC '10, pages 47–56, New York, NY, USA, 2010. ACM.

[LR99] F. T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46:787–832, 1999.

[Mad78] W. Mader. A reduction method for edge connectivity in graphs. *Ann. Discrete Math.*, 3:145–164, 1978.

[MM10] Konstantin Makarychev and Yury Makarychev. Metric extension operators, vertex sparsifiers and lipschitz extendability. In *FOCS*, pages 255–264. IEEE Computer Society, 2010.

[Moi09] Ankur Moitra. Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size. In *FOCS*, pages 3–12. IEEE Computer Society, 2009.

[MT10] Robin A. Moser and Gábor Tardos. A constructive proof of the general lovász local lemma. *J. ACM*, 57:11:1–11:15, February 2010.

[Räc02] Harald Räcke. Minimizing congestion in general networks. In *In Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 43–52, 2002.

[RS90] N. Robertson and P. D. Seymour. Outline of a disjoint paths algorithm. In *Paths, Flows and VLSI-Layout*. Springer-Verlag, 1990.

[RT87] Prabhakar Raghavan and Clark D. Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, December 1987.

[RZ10] Satish Rao and Shuheng Zhou. Edge disjoint paths in moderately connected graphs. *SIAM J. Comput.*, 39(5):1856–1887, 2010.

[SL07] Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In David S. Johnson and Uriel Feige, editors, *STOC*, pages 661–670. ACM, 2007.

[Sri97] Aravind Srinivasan. Improved approximations for edge-disjoint paths, unsplittable flow, and related routing problems. In *IEEE Symposium on Foundations of Computer Science*, pages 416–425, 1997.

[VV04] Kasturi R. Varadarajan and Ganesh Venkataraman. Graph decomposition and a greedy algorithm for edge-disjoint paths. In *Symposium on Discrete Algorithms*, pages 379–380, 2004.

A Table of Parameters

$\gamma_{\mathrm{KRV}}(k)$	$\Theta(\log^2 k)$	Parameter from the cut-matching game of [KRV06], from Theorem 5. Is also denoted by γ.
$\alpha_{\mathrm{ARV}}(k)$	$O(\sqrt{\log k})$	Approximation factor of the algorithm of [ARV09] for Sparsest Cut.
$\alpha(k)$	$\frac{1}{2^{11}\cdot\gamma_{\mathrm{KRV}}(k)\cdot\log k} = \Omega\left(\frac{1}{\log^3 k}\right)$	Well-linkedness parameter from Theorem 6.
$\alpha_{\mathrm{WL}}(k)$	$\alpha(k)/\alpha_{\mathrm{ARV}}(k) = \Omega\left(\frac{1}{\log^{3.5} k}\right)$	Well-linkedness parameter from Corollary 1.
$\beta(k)$	$\Theta(\log k)$	Flow-cut gap for concurrent flow on k terminals.
k_1	$\frac{k}{192\gamma^3\log\gamma} = \Omega\left(\frac{k}{\log^6 k \log\log k}\right)$	Parameter from the definition of legal contracted graphs.
p	$\frac{8\beta(k)}{\alpha_{\mathrm{WL}}(k)} = O(\log^{4.5} k)$	Grouping parameter for the sets Γ_j.
k'	$\frac{1}{2\gamma^3}\cdot\lfloor\frac{k_1}{6p}\rfloor = \Omega\left(\frac{k}{\log^{16.5} k\log\log k}\right)$	Number of vertices in the expander X.
k^*	$\lfloor\frac{k_1}{6p}\rfloor$	Size of sets Γ'_j (that contain at most one edge from each group of $\mathcal{G}_j$).

B Proofs Omitted from Section 2

B.1 Proof of Theorem 4

Let $\ell = 4d\beta(n)$, where $\beta(n) = O(\log n)$ is the flow-cut gap for undirected graphs. The algorithm greedily selects a source-sink pair (s_i, t_i) that has a path P of length at most ℓ connecting s_i to t_i in the current graph G. We then remove all vertices of P from the graph G and continue. The algorithm terminates when for each remaining source-sink pair (s_i, t_i), every path connecting s_i to t_i has length at least ℓ.

Note that in each iteration of the algorithm, we route one demand pair, and remove at most $(\ell+1)d$ edges from the graph. The key to the algorithm analysis is to show that when the algorithm terminates, we have removed many edges from the graph, and therefore we have routed many of the demand pairs.

Let E' be the subset of edges removed from the graph by the algorithm, and let E'' be the subset of remaining edges. We first claim that there is a multicut in graph G whose value is at most $|E'|+|E''|\cdot\beta(n)/\ell$. Indeed, let $G'=G[E'']$ be the graph obtained when the algorithm terminates, and let $\mathcal{M}'$ be the set of the surviving source-sink pairs. Consider the instance of the multicut problem on graph G' with the set $\mathcal{M}'$ of demand pairs. Setting the weight of each edge in E'' to $1/\ell$, we obtain a feasible fractional solution to this multicut instance, since the length of every path connecting every pair of terminals is at least ℓ. Therefore, there is an integral solution to this multicut instance of value $|E''|\cdot\beta(n)/\ell$. Adding the subset E' of edges, we obtain a feasible solution to the multicut problem on the original graph G of value $|E'|+|E''|\cdot\beta(n)/\ell$.

On the other hand, the value of any multicut on graph G is at least $|V|/4$. Indeed, if E^* is any feasible solution to the multicut problem, then each connected component C of $G\setminus E^*$ contains at most $|V|/2$ vertices, and therefore has at least $|V(C)|/2$ out-going edges. Since each edge is counted at most twice, we get that $|E^*|\geq |V|/4$.

We conclude that $|E'|+|E''|\cdot\beta(n)/\ell\geq |V|/4$, and so

$$|E'|\geq\frac{|V|}{4}-\frac{|E''|\cdot\beta(n)}{\ell}\geq\frac{|V|}{4}-\frac{|E|\cdot\beta(n)}{\ell}\geq\frac{|V|}{4}-\frac{d|V|\beta(n)}{2\ell}\geq\frac{|V|}{8}$$

since $\ell=4d\beta(n)$. Therefore, at least $|V|/8$ edges have been deleted from the graph. Since in each iteration we only delete at most $d(\ell+1)$ edges, overall the number of pairs routed is at least $\frac{|V|}{8d(\ell+1)}=\Omega\left(\frac{|V|}{d^2\log n}\right)$.

B.2 Proof of Theorem 7

Let T be the spanning tree of the graph G, and assume that it is rooted at some vertex r. We perform a number of iterations, where in each iteration we delete some edges and vertices from T. For each vertex v of the tree T, let T_v denote the sub-tree rooted at v, and let $w(T_v)$ denote the total weight of all vertices in T_v. We build the partition $\mathcal{G}$ of V gradually. At the beginning, $\mathcal{G}=\emptyset$. While $w(T)>3p$, we perform the following iteration:

- Let v be the lowest vertex in the tree T, such that $w(T_v)>p$.
- If $w(T_v)\leq 2p$, then we add a new group U to $\mathcal{G}$, containing all vertices of T_v, and we delete T_v from the tree T, setting $T_U=T_v$.
- Otherwise, let $u_1,\ldots,u_k$ be the children of v, and let j be the smallest index, such that $\sum_{i=1}^{j}w(T_{u_i})\geq p$. We add a new group U to $\mathcal{G}$, consisting of all vertices in trees $T_{u_1},\ldots,T_{u_j}$. Notice that $w(U)\leq 2p$ must hold. We let T_U be the sub-tree of T consisting of v and the trees $T_{u_1},\ldots,T_{u_j}$. We delete the trees $T_{u_1},\ldots,T_{u_j}$ from the tree T.

Notice that if, at the beginning of the current iteration, $w(T)>3p$, then at the end of the current iteration, $w(T)>p$ must hold. In the last iteration, when $w(T)\leq 3p$, we add a final group U to $\mathcal{G}$, containing all vertices currently in the tree T, and we let T_U be the current tree T. It is easy to verify that all conditions of the theorem hold for the final partition $\mathcal{G}$ of V.

C Proof of Theorem 8

For the proof of this theorem, we need a more general definition of flow well-linkedness, that was used in [CKS05]. Suppose we are given a graph $G = (V, E)$, and for each vertex $v \in V$, we are given a weight $\pi(v)$. For a subset $S \subseteq V$ of vertices, let $\pi(S) = \sum_{v \in S} \pi(v)$. We say that G is π-flow well-linked, iff each pair (u, v) of vertices can simultaneously send $\frac{\pi(u) \cdot \pi(v)}{\pi(V)}$ flow units to each other with no congestion. We start with the following theorem, that was proved in [CKS05], using a flow-well-linked graph decomposition.

Theorem 14 (Theorem 2.1 in [CKS05]) *Let $G = (V, E)$ be any graph and let $\mathcal{M}$ be a set of k source-sink pairs in G. We can efficiently find a partition $G_1, \ldots, G_\ell$ of G into vertex-disjoint induced subgraphs, and for each $1 \leq i \leq \ell$, find a weight function $\pi_i : V(G_i) \to \mathbb{R}^+$, with the following properties. Let $\mathcal{M}'_i \subseteq \mathcal{M}$ be the set of source-sink pairs contained in G_i, and let $\mathcal{T}'_i$ be the set of all terminals participating in $\mathcal{M}'_i$. Then:*

- *For all $1 \leq i \leq \ell$:*
  - *for all $u \in \mathcal{T}'_i$, $\pi_i(u) \leq 1$.*
  - *for all $(u, v) \in \mathcal{M}'_i$, $\pi_i(u) = \pi_i(v)$.*
  - *Graph G_i is π_i-flow well-linked.*
- $\sum_{i=1}^{\ell} \pi_i(\mathcal{T}'_i) = \Omega(\mathsf{OPT}/(\beta(k) \cdot \log \mathsf{OPT})) = \Omega(\mathsf{OPT}/\log^2 k)$.

In order to complete the proof of the theorem, it is enough to show that we can find, for each $1 \leq i \leq \ell$, a subset $\mathcal{M}_i \subseteq \mathcal{M}'_i$ of source-sink pairs, with $|\mathcal{M}_i| = \Omega(\pi_i(\mathcal{T}'_i))$, such that the set $\mathcal{T}_i$ of all terminals participating in pairs in $\mathcal{M}_i$ is flow-well-linked in G_i.

Fix some $1 \leq i \leq \ell$. We find a grouping $\mathcal{G}_i$ of the terminals in set $\mathcal{T}'_i$, using the weights π_i and the grouping parameter $p = 2$, as in Theorem 7, so for each group $U \in \mathcal{G}_i$, $2 \leq \pi_i(U) \leq 6$. Next, we will gradually construct the set $\mathcal{M}_i$ of source-sink pairs, starting from $\mathcal{M}_i = \emptyset$. In each iteration, we will add one source-sink pair to $\mathcal{M}_i$, and remove some source-sink pairs from $\mathcal{M}'_i$, charging their weights to the pair that was added to $\mathcal{M}_i$. While $\mathcal{M}'_i$ is non-empty, we perform the following procedure:

- Let $(s, t) \in \mathcal{M}'_i$ be any source-sink pair. Add (s, t) to $\mathcal{M}_i$.
- If both s and t belong to the same group $U \in \mathcal{G}$, then for each pair $(u, v) \in \mathcal{M}'_i$, where $u \in U$ or $v \in U$, remove (u, v) from $\mathcal{M}'_i$, and charge the weight $\pi_i(v)$ and $\pi_i(v)$ to (s, t). Notice that the total weight charged to (s, t) is at most 12.
- Otherwise, let U_1 be the group to which s belongs, and let U_2 be the group to which t belongs. For each pair $(u, v) \in \mathcal{M}'_i$, such that either $u \in U_1 \cup U_2$, or $v \in U_1 \cup U_2$, remove (u, v) from $\mathcal{M}'_i$, and charge the weights $\pi_i(u)$ and $\pi_i(v)$ to (s, t). Notice that the total weight charged to (s, t) in this step is at most 24.

The procedure stops when $\mathcal{M}'_i = \emptyset$. Let $\mathcal{M}_i$ be the resulting set of source-sink pairs, and let $\mathcal{T}_i$ be the set of terminals participating in them. From the above charging scheme, it is clear that $|\mathcal{M}_i| = \Omega(\pi_i(\mathcal{T}'_i))$, as required. Observe also that for each group $U \in \mathcal{G}_i$, at most one terminal $v \in U$ belongs to $\mathcal{T}_i$. Finally, we need to show that G_i is flow well-linked for $\mathcal{T}_i$. For each vertex $v \in \mathcal{T}_i$, let $U_v \in \mathcal{G}_i$ be the group to which v belongs.

Suppose we are given any matching $\mathcal{M}^*$ on the set $\mathcal{T}_i$ of terminals. We show how to route this matching with congestion at most 2 in G. We do so in two steps. In the first step, we construct a flow F_1, where for each pair $(v, v') \in \mathcal{M}^*$, the vertices in U_v send 1 flow unit in total to the vertices in $U_{v'}$, each vertex $x \in U_v$ sends at most $\pi_i(x)$ flow units and each vertex $y \in U_{v'}$ receives at most $\pi_i(y)$ flow units, with total congestion at most 1. This flow is defined as follows. Recall that graph G_i is π_i-well-linked. Therefore, every pair (x, y) of vertices can send $\frac{\pi_i(x)\cdot\pi_i(y)}{\pi_i(V(G_i))}$ flow units to each other with no congestion. Let F denote this flow. Fix some pair $(v, v') \in \mathcal{M}^*$. In flow F, there are $\pi_i(U_v)$ flow units originating from the vertices in U_v, that are then distributed among the vertices of G, and the amount of flow each vertex z of G receives is $\pi_i(z)\cdot\pi_i(U_v)/\pi_i(V(G_i))$. If $\pi_i(U_v) > 2$, we scale the flow originating from vertices in U_v down by factor $\pi_i(U_v)/2$, so that every vertex z of G now receives $2\pi_i(z)/\pi_i(V(G_i))$ flow units from U_v. We perform a similar transformation for the flow originating at the vertices of $U_{v'}$, and we concatenate both flows. As a result, we obtain a flow where the vertices in U_v send two flow units in total to the vertices in $U_{v'}$. Taking the union of these flows over all $(v, v') \in \mathcal{M}^*$, and scaling them down by factor 2, gives us the flow F_1. It is easy to see that the total congestion caused by F_1 is at most 1. This is since each flow-path in F is used at most twice: once for each of its end-points. Finally, in order to route the matching $\mathcal{M}^*$, consider any pair $(v, v') \in \mathcal{M}^*$. Vertex v will distribute one flow unit to the vertices in U_v, along the tree T_{U_v}, where the amount of flow each vertex $x \in T_{U_v}$ receives equals to the amount of flow it sends out in F_1. We then use the flow F_1 to route this one flow unit to the vertices of $U_{v'}$. Finally, vertex v' collects one flow unit from the vertices of $U_{v'}$ along the tree $T_{U_{v'}}$. It is easy to see that the total congestion caused by this flow is at most 2, since all trees $\{T_U\}_{U\in\mathcal{G}_i}$ are edge-disjoint.

D Proof of Theorem 13

We use the result of Leighton and Rao [LR99], who have shown that any demand that is routable on an expander graph with no congestion, can also be routed on relatively short paths with small congestion. Specifically, following is a slightly rephrased statement of Theorem 18 from [LR99], and its immediate corollary.

Theorem 15 (Theorem 18 from [LR99]) *Let G be any n-vertex α-expander with maximum vertex degree $d_{\max}$. Then every pair of vertices in G can send $\Omega(\alpha/(n\log n))$ flow units to each other with no congestion, on flow-paths of length $O(d_{\max}\log n/\alpha)$. Moreover, such flow can be found efficiently.*

Corollary 3 *Let G be any n-vertex α-expander with maximum vertex degree $d_{\max}$, and let M be any partial matching over the vertices of G. Then there is an efficient randomized algorithm that finds, for every pair $(u, v) \in M$, a set $\mathcal{P}_{u,v}$ of $h = \lceil\log n\rceil$ paths length $O(d_{\max}\log n/\alpha)$ each, such that the set $\mathcal{P} = \bigcup_{(u,v)\in M}\mathcal{P}_{u,v}$ of paths causes congestion $O(\log^2 n/\alpha)$ in G. The algorithm succeeds with high probability.*

Proof: We start by showing that there is a multi-commodity flow f, where every pair $(u, v) \in M$ of vertices sends one flow unit to each other simultaneously, on flow-paths of length $O(d_{\max}\log n/\alpha)$, with total congestion $O(\log n/\alpha)$. Let f' be the flow guaranteed by Theorem 15, scaled up by factor $O(\log n/\alpha)$, so that every pair of vertices now sends $1/n$ flow units to each other, with total congestion $O(\log n/\alpha)$. Let $(u, v) \in M$ be any pair of vertices. The flow between u and v is defined as follows: u will send $1/n$ flow units to each vertex of G, using the flow f', and v will collect $1/n$ flow units from each vertex in G, using the flow f'. In other words, the flow f between u and v is obtained by concatenating all flow-paths in f' originating at u with all flow-paths in f' terminating at v. It is easy to see then that every flow-path in f' is used at most twice: once by each of its endpoints; all flow-paths in f have length $O(d_{\max}\log n/\alpha)$, and the total congestion of flow f is $O(\log n/\alpha)$.

In order to find the sets $\mathcal{P}_{u,v}$ of paths for each pair $(u,v)\in M$, we perform the standard randomized rounding: for each pair $(u,v)\in M$, we perform h independent random trials. In each trial, we randomly choose one of the flow-paths connecting u to v, with probability equal to the amount of flow sent via this path in f, and add this path to $\mathcal{P}_{u,v}$. This ensures that all paths in set $\mathcal{P}_{u,v}$ have length $O(d_{\max}\log n/\alpha)$. The expected congestion on any edge is $O(h\log n/\alpha)$, and using the standard Chernoff bounds, it is easy to see that with high probability, the congestion on every edge is $O(\log^2 n/\alpha)$. □

We are now ready to prove Theorem 13.

Proof of Theorem 13 Let $L=O(d_{\max}\log n/\alpha)$ be the bound on the path length, and $\eta=O(\log^2 n/\alpha)$ the bound on the congestion guaranteed by Corollary 3. We set $m=\frac{(4e\cdot\eta^{c+2}\cdot d_{\max}^{c+2}\cdot L)^{1/c}}{h}=O\left(d_{\max}^{1+3/c}(\log n)^{1+5/c}/\alpha^{1+3/c}\right)$.

Given the matching $M\subseteq([r]\times[r])$, we assume w.l.o.g., that $M=\{(2i-1,2i)\}_{i=1}^{\lfloor r/2\rfloor}$. We extend the matching M to the vertices of G, by defining a matching M', as follows. For each $1\leq i\leq\lfloor r/2\rfloor$, we select any maximal matching M_i between the vertices of V_{2i-1} and the vertices of V_{2i}, and add it to M'. Notice that M_i must contain at least m pairs of vertices. Next, we use Corollary 3, to find a collection $\mathcal{P}$ of paths of length at most L each, that cause a total congestion of at most η in G, such that for each $(u,v)\in M'$, there is a set $\mathcal{P}_{u,v}$ of h paths connecting u to v in $\mathcal{P}$. For each $i:1\leq i\leq\lfloor r/2\rfloor$, let B_i denote the union of the sets $\mathcal{P}_{u,v}$ of paths, for $u\in V_{2i-1}$, $v\in V_{2i}$, and we call B_i the ith bundle. Notice that $|B_i|\geq mh$. If B_i contains more than mh paths, we discard paths from B_i, until $|B_i|=mh$ holds for all i.

Finally, we will select one path from each bundle, such that the resulting set of paths causes vertex congestion at most c. We do so, using the constructive version of the Lovasz Local Lemma by Moser and Tardos [MT10]. The next theorem summarizes the symmetric version of the result of [MT10].

Theorem 16 ([MT10]) *Let X be a finite set of mutually independent random variables in some probability space. Let $\mathcal{A}$ be a finite set of bad events determined by these variables. For each event $A\in\mathcal{A}$, let $vbl(A)\subseteq X$ be the unique minimal subset of variables determining A, and let $\Gamma(A)\subseteq\mathcal{A}$ be a subset of bad events B, such that $A\neq B$, but $vbl(A)\cap vbl(B)\neq\emptyset$. Assume further that for each $A\in\mathcal{A}$, $|\Gamma(A)|\leq D$, $\mathbf{Pr}[A]\leq p$, and $ep(D+1)\leq 1$. Then there is an efficient randomized algorithm that w.h.p. finds an assignment to the variables of X, such that none of the events in $\mathcal{A}$ holds.*

For each bundle B_i we choose one of its paths P_i independently at random. We let x_i be the variable indicating which path has been chosen. For each vertex $v\in V$, we let β_v be the bad event that v belongs to more than c of the chosen paths. Since the congestion on every edge due to paths in $\mathcal{P}$ is at most η, and the maximum vertex degree is $d_{\max}$, we get that there are at most $(\eta d_{\max})^{c+1}$ potential $(c+1)$-tuples of paths containing v (where we only consider $(c+1)$-tuples containing at most one path from each bundle), and each tuple is chosen with probability $1/(hm)^{c+1}$. Therefore, $\mathbf{Pr}[\beta_v]\leq\frac{(\eta d_{\max})^{c+1}}{(hm)^{c+1}}$. We denote $p=\left(\frac{\eta d_{\max}}{hm}\right)^{c+1}$.

The set $vbl(\beta_v)$ of variables contains all variables x_i, where the bundle B_i contains a path $P\in\mathcal{P}$, such that $v\in P$. Therefore, $|vbl(\beta_v)|\leq\eta d_{\max}$. For each such variable x_i, there are mh paths participating in the bundle B_i, each of which contains at most $(L+1)$ vertices. Therefore, $|\Gamma(\beta_v)|\leq mh(L+1)\eta d_{\max}$. We denote this value by D.

It now only remains to show that $(D+1)ep \leq 1$, which follows from the choice of $m = \frac{(4e \cdot \eta^{c+2} \cdot d_{\max}^{c+2} \cdot L)^{1/c}}{h}$.
□