Approximation Algorithms and Hardness of Integral Concurrent Flow

Parinya Chalermsook* Julia Chuzhoy † Alina Ene‡ Shi Li§

November 14, 2011

Abstract

We study an integral counterpart of the classical Maximum Concurrent Flow problem, that we call Integral Concurrent Flow (ICF). In the basic version of this problem (basic-ICF), we are given an undirected n-vertex graph G with edge capacities $c(e)$, a subset $\mathcal{T}$ of vertices called terminals, and a demand $D(t,t')$ for every pair (t,t') of the terminals. The goal is to find the maximum value λ, and a collection $\mathcal{P}$ of paths, such that every pair (t,t') of terminals is connected by $\lfloor \lambda D(t,t') \rfloor$ paths in $\mathcal{P}$, and the number of paths containing any edge e is at most $c(e)$. We show an algorithm that achieves a $\mathrm{poly}\log n$-approximation for basic-ICF, while violating the edge capacities by only a constant factor. We complement this result by proving that no efficient algorithm can achieve a factor α-approximation with congestion c for any values α, c satisfying $\alpha \cdot c = O(\log\log n/\log\log\log n)$, unless NP $\subseteq$ ZPTIME$(n^{\mathrm{poly}\log n})$.

We then turn to study the more general group version of the problem (group-ICF), in which we are given a collection $\{(S_1,T_1),\ldots,(S_k,T_k)\}$ of pairs of vertex subsets, and for each $1 \le i \le k$, a demand D_i is specified. The goal is to find the maximum value λ and a collection $\mathcal{P}$ of paths, such that for each i, at least $\lfloor \lambda D_i \rfloor$ paths connect the vertices of S_i to the vertices of T_i, while respecting the edge capacities. We show that for any $1 \le c \le O(\log\log n)$, no efficient algorithm can achieve a factor $O\left(n^{1/2^{2c+3}}\right)$-approximation with congestion c for the problem, unless NP $\subseteq$ DTIME$(n^{O(\log\log n)})$. On the other hand, we show an efficient randomized algorithm that finds $\mathrm{poly}\log n$-approximate solutions with constant congestion if we are guaranteed that the optimal solution contains at least $D \ge k\,\mathrm{poly}\log n$ paths connecting every pair (S_i,T_i).

*Department of Computer Science, University of Chicago, Chicago, IL 60615. Email: parinya@cs.uchicago.edu. Supported in part by NSF grant CCF-0844872

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

‡Dept. of Computer Science, University of Illinois, Urbana, IL 61801. Email: ene1@illinois.edu. Supported in part by NSF grants CCF-0728782, CCF-1016684 and CCF-0844872.

§Center for Computational Intractability, Department of Computer Science, Princeton University. Email: shili@cs.princeton.edu. Supported by NSF awards MSPA-MCS 0528414, CCF 0832797, AF 0916218 and CCF-0844872.

1 Introduction

Multicommodity flows are ubiquitous in computer science, and they are among the most basic and extensively studied combinatorial objects. Given an undirected n-vertex graph $G = (V, E)$ with capacities $c(e) > 0$ on edges $e \in E$, and a collection $\{(s_1, t_1), \ldots, (s_k, t_k)\}$ of source-sink pairs, two standard objective functions for multicommodity flows are: Maximum Multicommodity Flow, where the goal is to maximize the total amount of flow routed between the source-sink pairs, and Maximum Concurrent Flow, where the goal is to maximize a value λ, such that λ flow units can be simultaneously sent between every pair (s_i, t_i).

Many applications require however that the routing of the demand pairs is *integral*, that is, the amount of flow sent on each flow-path is integral. The integral counterpart of Maximum Multicommodity Flow is the Edge Disjoint Paths problem (EDP), where the goal is to find a maximum-cardinality collection $\mathcal{P}$ of paths connecting the source-sink pairs with no congestion. It is a standard practice to define the EDP problem on graphs with unit edge capacities, so a congestion of any solution $\mathcal{P}$ is the maximum number of paths in $\mathcal{P}$ sharing an edge. EDP is a classical routing problem that has been studied extensively. Robertson and Seymour [RS90] have shown an efficient algorithm for EDP when the number k of the demand pairs is bounded by a constant, but the problem is NP-hard for general values of k [Kar72]. The best currently known approximation algorithm, due to Chekuri, Khanna and Shepherd [CKS06], achieves an $O(\sqrt{n})$-approximation. The problem is also known to be $\Omega(\log^{1/2-\epsilon} n)$-hard to approximate for any constant ϵ, unless NP $\subseteq$ ZPTIME$(n^{\text{poly}\log n})$ [AZ05, ACG+10]. Moreover, the ratio of the Maximum Multicommodity Flow to the value of the optimal solution to the EDP problem instance can be as large as $\Omega(\sqrt{n})$ [CKS06]. Interestingly, when the value of the global minimum cut in G is $\Omega(\log^5 n)$, Rao and Zhou [RZ10] have shown a factor $\text{poly}\log n$-approximation algorithm for EDP, by rounding the multicommodity flow. Much better results are known if we slightly relax the problem requirements by allowing a small congestion. In a recent breakthrough, Andrews [And10] has shown an efficient randomized algorithm that w.h.p. routes $\Omega(\mathsf{OPT}/\text{poly}\log n)$ of the demand pairs with congestion $\text{poly}(\log\log n)$, where OPT is the value of the optimal solution with no congestion for the given EDP instance, and Chuzhoy [Chu11] has shown an efficient randomized algorithm that w.h.p. routes $\Omega(\mathsf{OPT}/\text{poly}\log k)$ of the demand pairs with a constant congestion. In fact the number of demand pairs routed by the latter algorithm is within a $\text{poly}\log k$-factor of the Maximum Multicommodity Flow value.

Assume now that we are given an instance where every demand pair (s_i, t_i) can simultaneously send D flow units to each other with no congestion. The algorithm of [Chu11] will then produce a collection $\mathcal{P}$ of $\Omega(Dk/\text{poly}\log k)$ paths connecting the demand pairs, but it is possible that some pairs are connected by many paths, while some pairs have no paths connecting them. In some applications however, it is important to ensure that **every** demand pair is connected by many paths.

In this paper, we propose to study an integral counterpart of Maximum Concurrent Flow, called Integral Concurrent Flow (ICF). We study two versions of ICF. In the more simple basic version (basic-ICF), we are given an undirected n-vertex graph $G = (V, E)$ with non-negative capacities $c(e)$ on edges $e \in E$, a subset $\mathcal{T} \subseteq V$ of k vertices called terminals, and a set $\mathcal{D}$ of demands over the terminals, where for each pair $(t_i, t_j) \in \mathcal{T}$, a demand $D(t_i, t_j)$ is specified. The goal is to find a maximum value λ, and a collection $\mathcal{P}$ of paths, such that for each pair (t_i, t_j) of terminals, set $\mathcal{P}$ contains at least $\lfloor \lambda \cdot D(t_i, t_j) \rfloor$ paths connecting t_i to t_j, and for each edge $e \in E$, at most $c(e)$ paths in $\mathcal{P}$ contain e.

The second and the more general version of the ICF problem that we consider is the group version (group-ICF), in which we are given an undirected n-vertex graph $G = (V, E)$ with edge capacities $c(e) > 0$, and k pairs of vertex subsets $((S_1, T_1), \ldots, (S_k, T_k))$. For each pair (S_i, T_i), we are also given a demand D_i. The goal is to find a maximum value λ, and a collection $\mathcal{P}$ of paths, such that for each $1 \leq i \leq k$, there are at least $\lfloor \lambda D_i \rfloor$ paths connecting the vertices of S_i to the vertices of T_i in $\mathcal{P}$, and every edge $e \in E$ belongs to at most $c(e)$ paths. It is easy to see that group-ICF generalizes both the basic-ICF and the EDP problems[1]. As in the EDP problem, we will sometimes relax the capacity constraints, and will instead only require that the maximum edge

[1]To reduce EDP to group-ICF, make D disjoint copies of the EDP instance. For each $1 \leq i \leq k$, let S_i contain all copies of s_i and T_i contain all copies of t_i. If we can find λD paths for every group (S_i, T_i), then some copy of the EDP instance will contain a solution of value at least λk.

congestion - the ratio of the number of paths containing the edge to its capacity - is bounded. We say that a set $\mathcal{P}$ of paths is a solution of value λ and congestion η, iff for every $1 \leq i \leq k$, at least $\lfloor \lambda \cdot D_i \rfloor$ paths connect the vertices of S_i to the vertices of T_i, and every edge $e \in E$ participates in at most $\eta \cdot c(e)$ paths in $\mathcal{P}$. Throughout the paper, we denote by λ^* the value of the optimal solution to the ICF instance, when no congestion is allowed. We say that a solution $\mathcal{P}$ is an α-approximation with congestion η iff for each $1 \leq i \leq k$, at least $\lfloor \lambda^* \cdot D_i/\alpha \rfloor$ paths connect the vertices of S_i to the vertices of T_i, and the congestion due to paths in $\mathcal{P}$ is at most η.

Given a multicommodity flow F, we say that it is a *fractional* solution of value λ to the group-ICF instance, iff for each demand pair (S_i, T_i), at least $\lambda \cdot D_i$ flow units are sent from the vertices of S_i to the vertices of T_i with no congestion. Throughout the paper, we denote by λ_{OPT} the value of the optimal fractional solution to the ICF problem instance. Observe that for basic-ICF, finding the optimal fractional solution is equivalent to solving the Maximum Concurrent Flow problem.

In addition to designing approximation algorithms for the ICF problem, an interesting question is the relationship between the optimal fractional and the optimal integral solutions for ICF. For example, suppose we are given a multicommodity flow, where for each $1 \leq i \leq k$, the vertices of S_i send D flow units to the vertices of T_i simultaneously with no congestion. What is the maximum value λ, for which we can find an integral solution where for each pair (S_i, T_i) at least $\lfloor \lambda D \rfloor$ paths connect the vertices of S_i to the vertices of T_i?

We start by showing an efficient randomized algorithm for basic-ICF that w.h.p. produces a solution of value $\lambda_{\text{OPT}}/\operatorname{poly}\log n$ and constant congestion. We also show that for any values η, α, such that $\eta \cdot \alpha \leq O(\log\log n/\log\log\log n)$, no efficient algorithm can find an α-approximate solution with congestion η to basic-ICF unless NP $\subseteq$ ZPTIME$(n^{\operatorname{poly}\log n})$. We then turn to the more challenging group-ICF problem. It is easy to see that when no congestion is allowed, the ratio of the optimal fractional to the optimal integral solution can be as large as $\Omega(\sqrt{n})$ for group-ICF, even when $k = 2$. Moreover, even if we allow congestion $c-1$, this ratio can still be as large as $\Omega(n^{1/c})$ (see Section B). We show that for any $0 < \eta \leq O(\log\log n)$ and $\alpha = O\left(n^{1/2^{2\eta+3}}\right)$, no efficient algorithm can find α-approximate solutions with congestion η for group-ICF unless NP $\subseteq$ DTIME$(n^{O(\log\log n)})$. Given an optimal integral solution $\mathcal{P}$ to the group-ICF problem instance, let $D = \min_i \{\lfloor \lambda^* \cdot D_i \rfloor\}$ be the minimum number of paths connecting any pair (S_i, T_i) in this solution. Our hardness result only holds for the regime where $D < k$. We show that if $D > k \operatorname{poly}\log n$, then there is an efficient algorithm that finds a $(\operatorname{poly}\log n)$-approximate solution with constant congestion. The value of the solution is in fact $\lambda_{\text{OPT}}/\operatorname{poly}\log n$, where λ_{OPT} is the value of the optimal fractional solution. Therefore, when we allow a constant congestion, the ratio of the optimal fractional to the optimal integral solution becomes only polylogarithmic if $D > k \operatorname{poly}\log n$.

Our Results and Techniques Our first result is an approximation algorithm for the basic-ICF problem.

Theorem 1 *There is an efficient randomized algorithm, that, given any instance of* basic-ICF*, w.h.p. produces an integral solution of value $\lambda_{OPT}/\operatorname{poly}\log n$ and constant congestion, where λ_{OPT} is the value of the optimal fractional solution with no congestion to the* ICF *instance.*

The main technical tool that our algorithm uses is a graph decomposition similar to the one proposed by Andrews [And10]. Assume first that the value of the minimum cut in graph G is polylogarithmic. We can then define $\operatorname{poly}\log n$ new graphs $G_1, \ldots, G_r$, where for each $1 \leq j \leq r$, $V(G_j) = V(G)$, and the edges in graphs G_j form a partition of the edges in G. If the value of the minimum cut in G is large enough, we can furthermore ensure that the value of the minimum cut in each resulting graph G_j is $\Omega(\log^5 n)$. We can then use the algorithm of Rao and Zhou [RZ10] to find $\lambda^* \cdot \sum_i D_i/\operatorname{poly}\log n$ paths connecting the source-sink pairs in each graph G_j separately. By appropriately choosing the subsets of source-sink pairs for each graph G_j to connect, we can obtain a polylogarithmic approximation for the basic-ICF problem instance.

Unfortunately, it is possible that the global minimum cut in graph G is small. Andrews [And10] in his paper on the EDP problem, suggested to get around this difficulty as follows. Let $L = \operatorname{poly}\log n$ be a parameter. For any subset C of vertices in G, let $\operatorname{out}(C) = E(C, V \setminus C)$. We say that a subset C of vertices is a large cluster iff $|\operatorname{out}(C)| \geq L$, and otherwise we say that it is a small cluster. Informally, we say that C has the

bandwidth property iff we can send $1/|\operatorname{out}(C)|$ flow units between every pair of edges in $\operatorname{out}(C)$ with small congestion inside the cluster C. Finally, we say that C is a critical cluster iff it is a large cluster, and we are given a partition $\pi(C)$ of its vertices into small clusters, such that on the one hand, each cluster in $\pi(C)$ has the bandwidth property, and on the other hand, the graph obtained from $G[C]$ by contracting every cluster in $\pi(C)$ is an expander. The key observation is that if C is a critical cluster, then we can integrally route demands on the edges of $\operatorname{out}(C)$ inside C, by using standard algorithms for routing on expanders. The idea of Andrews is that we can use the critical clusters to "hide" the small clusters in G.

More specifically, the graph decomposition procedure of Andrews consists of two steps. In the first step, he constructs what we call a Q-J decomposition $(\mathcal{Q}, \mathcal{J})$ of graph G. Here, $\mathcal{Q}$ is a collection of disjoint critical clusters and $\mathcal{J}$ is a collection of disjoint small clusters that have the bandwidth property, and $\mathcal{Q} \cup \mathcal{J}$ is a partition of $V(G)$. This partition ensures that every cut separating any pair of clusters in $\mathcal{Q}$ is large, containing at least $\operatorname{poly}\log n$ edges, and moreover we can connect all edges in $\bigcup_{C \in \mathcal{J}} \operatorname{out}(C)$ to the edges of $\bigcup_{C \in \mathcal{Q}} \operatorname{out}(C)$ by paths that together only cause a small congestion.

Given a Q-J decomposition $(\mathcal{Q}, \mathcal{J})$, Andrews then constructs a new graph H, whose vertices are $\{v_Q \mid Q \in \mathcal{Q}\}$, and every edge $e = (v_Q, v_{Q'})$ in H is mapped to a path P_e in G connecting some vertex of Q to some vertex of Q', such that the total congestion caused by the set $\{P_e \mid e \in E\}$ of paths in graph G is small. Moreover, graph H preserves, to within a polylogarithmic factor, all cuts separating the clusters of $\mathcal{Q}$ in graph G. In particular, the size of the global minimum cut in H is large, and any integral routing in graph H can be transformed into an integral routing in G. This reduces the original problem to the problem of routing in the new graph H. Since the size of the minimum cut in graph H is large, we can now apply the algorithm proposed above to graph H.

We revisit the Q-J decomposition and the construction of the graph H from [And10], and obtain an improved construction with stronger parameters. In particular, it allows us to reduce the routing congestion to constant, and to reduce the powers of the logarithms in the construction parameters. The Q-J decomposition procedure of [And10] uses the graph tree decomposition of Räcke [Räc02] as a black box. We instead perform the decomposition directly, thus improving some of its parameters. We also design a new well-linked decomposition procedure that may be of independent interest.

Our next result shows that basic-ICF is hard to approximate, using a simple reduction from the Congestion Minimization problem.

Theorem 2 *Given an n-vertex graph G with unit edge capacities, a collection $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of source-sink pairs, and integers c, D, such that $Dc \leq O(\log\log n / \log\log\log n)$, no efficient algorithm can distinguish between the following two cases unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\operatorname{poly}\log n})$: (i) There is a collection $\mathcal{P}$ of paths that causes congestion 1, with D paths connecting s_i to t_i for each $1 \leq i \leq k$; and (ii) Any collection $\mathcal{P}$ of paths, containing, for each $1 \leq i \leq k$, at least one path connecting s_i to t_i, causes congestion at least c.*

We then turn to the group-ICF problem, and prove that it is hard to approximate in the following theorem.

Theorem 3 *Assume that we are given an n-vertex graph $G = (V, E)$ with unit edge capacities, and a collection $(S_1, T_1), \ldots, (S_k, T_k)$ of pairs of vertex subsets. Let c be any integer, $0 < c \leq O(\log\log n)$ and let $D = O\left(n^{1/2^{2c+3}}\right)$. Then unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{O(\log\log n)})$, no efficient algorithm can distinguish between the following two cases: (i) There is a collection $\mathcal{P}^*$ of paths that causes congestion 1, and contains, for every $1 \leq i \leq k$, D paths connecting the vertices of S_i to the vertices of T_i; and (ii) Any set $\mathcal{P}^*$ of paths, containing, for each $1 \leq i \leq k$, at least one path connecting a vertex of S_i to a vertex of T_i, causes congestion at least c.*

The proof of Theorem 3 establishes a connection between group-ICF and the Machine Minimization Scheduling problem, and then follows the hardness of approximation proof of [CN06] for the scheduling problem. Finally, we show an approximation algorithm for the group-ICF problem.

Theorem 4 *Suppose we are given an instance of group-ICF, and let $D = \min_i \{\lambda_{OPT} \cdot D_i\}$ be the minimum amount of flow sent between any pair (S_i, T_i) in the optimal fractional solution. Assume further that $D \geq \Delta'$, where $\Delta' = k \operatorname{poly}\log n$ is a parameter whose value we set later. Then there is an efficient randomized algorithm that finds a solution of value $\lambda_{OPT} / \operatorname{poly}\log n$ with constant congestion for the group-ICF instance.*

We now give a high-level overview of the proof of Theorem 4. We first define two special cases of the problem. The first special case is called a split instance, where we are given a partition $\mathcal{C}$ of the vertices in $V(G)$ into clusters, such that, in the optimal fractional solution, each flow-path is contained in some cluster $C \in \mathcal{C}$, and moreover, for each $1 \leq i \leq k$, the amount of flow sent between the vertices of S_i and the vertices of T_i inside any cluster $C \in \mathcal{C}$ is at most $\lambda_{\text{OPT}} \cdot D_i / \operatorname{poly}\log n$. We show an algorithm to solve split instances using the algorithm of [Chu11] for EDP and a simple randomized rounding procedure. The second special case includes instances for which we can find good Q-J decomposition. We say that a Q-J decomposition is good, iff no flow-path in the optimal fractional solution is contained in any cluster in $\mathcal{X}$, where $\mathcal{X} = \mathcal{J} \cup \left(\bigcup_{Q \in \mathcal{Q}} \pi(Q)\right)$, and $\mathcal{Q} \neq \emptyset$. We show an algorithm to find a polylogarithmic approximation with constant congestion for instances where a good Q-J decomposition is given. This algorithm is similar to the algorithm from Theorem 1 for basic-ICF. Finally, given any instance of the group-ICF problem, we partition it into a number of sub-instances, such that each edge of G only belongs to a constant number of sub-instances, and each sub-instance is either a split instance, or we are able to find a good Q-J decomposition for it.

Organization. We start with Prelimiaries in Section 2, and present the improved Q-J decomposition together with the construction of the graph H in Section 3. We show an algorithm for basic-ICF in Section 4, and hardness of basic-ICF and group-ICF in Sections 5 and 6 respectively. An algorithm for group-ICF appears in Section 7. For convenience, a list of the main parameters is given in Section A of the Appendix.

2 Preliminaries

In all our algorithmic results, we first solve the problem for the special case where all edge capacities are unit, and then extend our algorithms to general edge capacities. Therefore, in this section, we only discuss graphs with unit edge capacities.

Given any subset $S \subseteq V$ of vertices in graph G, let $\operatorname{out}_G(S) = E_G(S, V \setminus S)$. We omit the subscript G when clear from context. Let $\mathcal{P}$ be any collection of paths in graph G. We say that paths in $\mathcal{P}$ cause congestion η, iff for each edge $e \in E(G)$, the number of paths in $\mathcal{P}$ containing e is at most η.

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

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

Given any two subsets V_1, V_2 of vertices, we denote by $F : V_1 \rightsquigarrow_\eta V_2$ a flow from the vertices of V_1 to the vertices of V_2 where each vertex in V_1 sends one flow unit, and the congestion due to F is at most η. Similarly, we denote by $\mathcal{P} : V_1 \rightsquigarrow_\eta V_2$ a collection of paths $\mathcal{P} = \{P_v \mid v \in V_1\}$, where each path P_v originates at v and terminates at some vertex of V_2, and the paths in $\mathcal{P}$ cause congestion at most η. We define flows and path sets between subsets of edges similarly. For example, given two collections E_1, E_2 of edges of G, we denote by $F : E_1 \rightsquigarrow_\eta E_2$ a flow that causes congestion at most η in G, where each flow-path has an edge in E_1 as its first edge, and an edge in E_2 as its last edge, and moreover each edge in E_1 sends one flow unit (notice that it is then guaranteed that each edge in E_2 receives at most η flow units due to the bound on congestion). We will often be interested in a scenario where we are given a subset $S \subseteq V(G)$ of vertices, and $E_1, E_2 \subseteq \operatorname{out}(S)$. In this case, we say that a flow $F : E_1 \rightsquigarrow_\eta E_2$ is *contained* in S, iff for each flow-path P in F, all edges of P belong to $G[S]$, except for the first and the last edges that belong to $\operatorname{out}(S)$. Similarly, we say that a set $\mathcal{P} : E_1 \rightsquigarrow_\eta E_2$ of paths is contained in S, iff all inner edges on paths in $\mathcal{P}$ belong to $G[S]$.

Edge-Disjoint Paths. We use the algorithm of [Chu11] for EDP, summarized in the following theorem.

Theorem 5 ([Chu11]) *Let G be any graph with unit edge capacities and a set $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of source-sink pairs. Assume further that there is a multicommodity flow where the pairs in $\mathcal{M}$ altogether send* OPT *flow units to each other, with no congestion, and at most one flow unit is sent between each pair. Then there is an efficient randomized algorithm that w.h.p. finds a collection $\mathcal{P}$ of paths, connecting at least* $\mathsf{OPT}/\alpha_{\text{EDP}}$ *of the demand pairs, such that the congestion of $\mathcal{P}$ is at most $\eta_{\text{EDP}} = 14$, where $\alpha_{\text{EDP}} = \operatorname{poly}\log k$.*

Sparsest Cut, Flow-Cut Gap and Well-Linkedness. Given a graph $G = (V, E)$ with a subset $\mathcal{T}$ of vertices called terminals, the *sparsity* of any partition (A, B) of V is $\frac{|E(A,B)|}{\min\{|A\cap\mathcal{T}|,|B\cap\mathcal{T}|\}}$. The goal of the sparsest cut problem is to find a partition (A, B) of V with minimum sparsity. Arora, Rao and Vazirani [ARV09] have shown an $O(\sqrt{\log k})$-approximation algorithm for the sparsest cut problem. We denote by $\mathcal{A}_{\text{ARV}}$ this algorithm and by $\alpha_{\text{ARV}}(k) = O(\sqrt{\log k})$ its approximation factor.

Sparsest cut is the dual of the Maximum Concurrent Flow problem, where for each pair (t, t') of terminals, the demand $D(t, t') = 1/k$. The maximum possible ratio, in any graph, between the value of the minimum sparsest cut and the value λ of the maximum concurrent flow, is called the *flow-cut gap*. The flow-cut gap in undirected graphs, that we denote by $\beta_{\text{FCG}}(k)$ throughout the paper, is $\Theta(\log k)$ [LR99, GVY95, LLR94, AR98]. In particular, if the value of the sparsest cut in graph G is α, then every pair of terminals can send at least $\frac{\alpha}{k\beta_{\text{FCG}}(k)}$ flow units to each other simultaneously with no congestion. Moreover, any 1-restricted set $\mathcal{D}$ of demands on the set $\mathcal{T}$ of terminals can be routed with congestion at most $2\beta_{\text{FCG}}(k)/\alpha$ in G.

Given any subset $S \subseteq V$ of vertices, we say that S is *α-well-linked*, iff for any partition (A, B) of S, if we denote $\mathcal{T}_A = \operatorname{out}(S) \cap \operatorname{out}(A)$ and $\mathcal{T}_B = \operatorname{out}(S) \cap \operatorname{out}(B)$, then $|E(A, B)| \geq \alpha \cdot \min\{|\mathcal{T}_A|, |\mathcal{T}_B|\}$.

Given a subset S of vertices, we can sub-divide every edge $e \in \operatorname{out}(S)$ by a vertex z_e, and set $\mathcal{T}' = \{z_e \mid e \in \operatorname{out}(S)\}$. Let G_S be the sub-graph of the resulting graph induced by $S \cup \mathcal{T}'$. Then S is α-well-linked in G iff the value of the sparsest cut in graph G_S for the set $\mathcal{T}'$ of terminals is at least α. In particular, if S is α-well-linked, then any 1-restricted set $\mathcal{D}$ of demands on $\operatorname{out}(S)$ can be fractionally routed inside S with congestion at most $2\beta_{\text{FCG}}(k)/\alpha$.

Similarly, given any graph $G = (V, E)$ with a subset $\mathcal{T}$ of vertices called terminals, we say that G is α-well-linked for $\mathcal{T}$ iff for any partition (A, B) of V, $|E_G(A, B)| \geq \alpha \cdot \min\{|\mathcal{T} \cap A|, |\mathcal{T} \cap B|\}$.

Let $L = \operatorname{poly}\log n$ be a parameter to be fixed later. (We use different values of L in different algorithms) We say that a cluster $X \subseteq V(G)$ is *small* iff $|\operatorname{out}(X)| \leq L$, and we say that it is *large* otherwise.

A useful tool in graph routing algorithms is a well-linked decomposition [CKS05, Räc02]. This is a procedure that, given any subset S of vertices, produces a partition $\mathcal{W}$ of S into well-linked subsets. In the following theorem we describe a new well-linked decomposition. In addition to the standard properties guaranteed by well-linked decompositions, we obtain a collection of paths connecting the edges of $\bigcup_{R\in\mathcal{W}} \operatorname{out}(R)$ to the edges of $\operatorname{out}(S)$, with a small congestion. We defer the proof of the theorem to the Appendix.

Theorem 6 (Extended well-linked decomposition) *There is an efficient algorithm, that, given a set S of vertices, with $|\operatorname{out}(S)| = k'$, produces a partition $\mathcal{W}$ of S with the following properties.*

- *For each set $R \in \mathcal{W}$, $|\operatorname{out}(R)| \leq k'$. If R is a large cluster, then it is $\alpha_W = \Omega\left(1/\log^{1.5} n\right)$-well-linked. If R is a small cluster, then it is $\alpha_S = \Omega\left(1/(\log\log n)^{1.5}\right)$-well-linked.*
- *Let $E^* = \left(\bigcup_{R\in\mathcal{W}} \operatorname{out}(R)\right) \setminus \operatorname{out}(S)$. Then we can efficiently find a set $N = \{\tau_e \mid e \in E^*\}$ of paths called* tendrils *contained in $G[S]$, where tendril τ_e connects edge e to some edge of $\operatorname{out}(S)$, each edge in $\operatorname{out}(S)$ participates in at most one tendril, and the total congestion caused by N is at most 3.*
- $|E^*| \leq 0.4|\operatorname{out}(S)|$.

Bandwidth Property and Critical Clusters. Given a graph G, and a subset S of vertices of G, with $|\operatorname{out}(S)| = k'$, we say that the *modified bandwidth condition* holds for S, iff S is α_{BW}-well-linked if it is a large cluster, and it is α_S-well-linked if it is a small cluster, where $\alpha_S = \Omega\left(1/(\log\log n)^{1.5}\right)$, and $\alpha_{\text{BW}} = \alpha_W \cdot \alpha_S = \Omega\left(\frac{1}{(\log n \log\log n)^{1.5}}\right)$. For simplicity, we will use "bandwidth property" instead of "modified bandwidth property" from now on.

Given a subset S of vertices of G, and a partition π of S, let H_S be the following graph: start with $G[S]$, and contract each cluster $C \in \pi$ into a super-node v_C. Set the weight $w(v_C)$ of v_C to be $|\operatorname{out}_G(C)|$ (notice that the weight takes into account all edges incident on C, including those in $\operatorname{out}(S)$). We use the parameter $\lambda = \frac{\alpha_{\mathrm{BW}}}{8\alpha_{\mathrm{ARV}}(n)} = \Omega\left(\frac{1}{\log^2 n \cdot (\log\log n)^{1.5}}\right)$.

Definition 1 *Given a subset S of vertices of G, with $|\operatorname{out}(S)| = k'$, and a partition π of S, we say that (S, π) has the weight property iff for any partition (A, B) of $V(H_S)$, $|E_{H_S}(A, B)| \geq \lambda \cdot \min\left\{\sum_{v \in A} w(v), \sum_{v \in B} w(v)\right\}$.*

Definition 2 *Given a subset S of vertices and a partition π of S, we say that S is a* critical cluster *iff (1) S is a large cluster and it has the bandwidth property; (2) Every cluster $R \in \pi$ is a small cluster and it has the bandwidth property; and (3) (S, π) has the weight property. Additionally, if $S = \{v\}$, and the degree of v is greater than L, then we also say that S is a critical cluster.*

Let $\eta^* = \frac{2\beta_{\mathrm{FCG}}(L)}{\alpha_S} = O((\log\log n)^{2.5})$. We say that a cut $(S, \overline{S})$ in G is *large* iff $|E(S, \overline{S})| \geq \frac{L}{4\eta^*}$. Note that we somewhat abuse the notation: We will say that a cluster S is large iff $|\operatorname{out}(S)| > L$, but we say that a cut $(S, \overline{S})$ is large iff $|E(S, \overline{S})| \geq \frac{L}{4\eta^*}$

In the next lemma we show that if we are given any large cluster S, then we can find a critical sub-cluster Q of S. Moreover, there is a subset of at least $L/4$ edges of $\operatorname{out}(Q)$ that can be routed to the edges of $\operatorname{out}(S)$. One can prove a similar lemma using the Räcke decomposition as a black-box. Since we use slightly different parameters in the definitions of small and critical clusters, we prove it directly in the Appendix.

Lemma 1 *Let S be any large cluster that has the bandwidth property. Then we can efficiently find a critical cluster $Q \subseteq S$, a subset $E_Q \subseteq \operatorname{out}(Q)$ of $L/4$ edges, and a set $\mathcal{P}_Q : E_Q \rightsquigarrow_{\eta^*} \operatorname{out}(S)$ of paths, which are contained in $S \setminus Q$, such that for each edge $e \in \operatorname{out}(S)$, at most one path of $\mathcal{P}_Q$ terminates at e.*

Suppose we are given a collection $\mathcal{C}$ of disjoint vertex subsets in graph G. We say that a cut (A, B) in graph G is *canonical* w.r.t. $\mathcal{C}$, iff for each $C \in \mathcal{C}$, either $C \subseteq A$, or $C \subseteq B$. We say that is is a non-trivial canonical cut, iff both A and B contain at least one cluster in $\mathcal{C}$.

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

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 for each group $U \in \mathcal{G}$, $p \leq w(U) \leq 3p$, where $w(U) = \sum_{v \in U} w(v)$, and the trees $\{T_U\}_{U \in \mathcal{G}}$ are edge-disjoint.*

Routing on Small and Critical Clusters. We use the following theorem from [Chu11].

Theorem 8 *Let G be any graph and $\mathcal{T}$ any subset of k vertices called terminals, such that G is α-well-linked for $\mathcal{T}$. Then we can efficiently find a partition $\mathcal{G}$ of the terminals in $\mathcal{T}$ into groups of size $\frac{\operatorname{poly}\log k}{\alpha}$, such that, for any $(1, \mathcal{G})$-restricted set $\mathcal{D}$ of demands on $\mathcal{T}$, there is an efficient randomized algorithm that w.h.p. finds an integral routing of $\mathcal{D}$ in G with edge congestion at most 15.*

Suppose we are given a small cluster S that has the bandwidth property. Since $|\operatorname{out}(S)| \leq \operatorname{poly}\log n$, and S is α_S-well-linked, we can use Theorem 8 to find a partition $\mathcal{G}(S)$ of the edges of S into sets of size $\operatorname{poly}\log\log n$, such that any $(1, \mathcal{G}(S))$-restricted set $\mathcal{D}$ of demands can be integrally routed inside S with congestion 15 w.h.p. We also use the following simple observation, whose proof appears in Appendix.

Observation 1 *Let $\mathcal{G}$ be any partition of the set $\mathcal{T}$ of terminals, and let $\mathcal{D}$ be any set of $(\gamma, \mathcal{G})$-restricted integral demands. Then we can efficiently find 4γ sets $\mathcal{D}_1, \ldots, \mathcal{D}_{4\gamma}$ of $(1, \mathcal{G})$-restricted integral demands, such that any routing of the demands in set $\bigcup_{i=1}^{4\gamma} \mathcal{D}_i$ gives a routing of the demands in $\mathcal{D}$ with the same congestion, and moreover, if the former routing is integral, so is the latter.*

Combining Theorem 8 with Observation 1, we get the following corollary for routing across small clusters.

Corollary 1 *Given any small cluster S that has the bandwidth property, we can efficiently find a partition $\mathcal{G}_S$ of the edges of $\mathrm{out}(S)$ into groups of size at most $z = \mathrm{poly}\log\log n$, such that, for any $\gamma \geq 1$, given any $(\gamma, \mathcal{G})$-restricted set $\mathcal{D}$ of demands on the edges of $\mathrm{out}(S)$, there is an efficient randomized algorithm, that w.h.p. finds an integral routing of $\mathcal{D}$ inside $G[S]$ with congestion at most 60γ.*

The following theorem, whose proof appears in Appendix, gives an efficient algorithm for integral routing across critical clusters.

Theorem 9 *Suppose we are given any cluster S, together with a partition π of S into small clusters, such that every cluster $C \in \pi$ has the bandwidth property, and (S, π) has the weight property. Then we can efficiently find a partition $\mathcal{G}$ of the edges of $\mathrm{out}(S)$ into groups of size at least $Z = O(\log^4 n)$ and at most $3Z$, such that, for any set $\mathcal{D}$ of $(1, \mathcal{G})$-restricted demands on $\mathrm{out}(S)$, there is an efficient randomized algorithm that w.h.p. routes $\mathcal{D}$ integrally in $G[S]$ with congestion at most 721.*

3 Graph Decomposition and Splitting

In this section we present the main technical tools that our algorithms use: the Q-J decomposition, the construction of the graph H, and the splitting of H into sub-graphs. We start with the Q-J decomposition. We assume that we are given a non-empty collection $\mathcal{Q}_0$ of disjoint critical clusters in graph G, with the following property: if (A, B) is any non-trivial canonical cut in graph G w.r.t. $\mathcal{Q}_0$, then it is a large cut. For motivation, consider the basic-ICF problem, and assume that every cut separating the terminals in $\mathcal{T}$ is a large cut. Then we can set $\mathcal{Q}_0 = \{\{t\} \mid t \in \mathcal{T}\}$. For the group-ICF problem, we will compute $\mathcal{Q}_0$ differently, by setting $\mathcal{Q}_0 = \{Q\}$ where Q is an arbitrary critical cluster in G.

Suppose we are given a collection $\mathcal{Q}$ of disjoint critical clusters, and a collection $\mathcal{J}$ of disjoint small clusters, such that $\mathcal{Q} \cup \mathcal{J}$ is a partition of $V(G)$. Let $E^Q = \bigcup_{Q \in \mathcal{Q}} \mathrm{out}(Q)$, and let $E^J = \left(\bigcup_{J \in \mathcal{J}} \mathrm{out}(J)\right) \setminus E^Q$. We say that $(\mathcal{Q}, \mathcal{J})$ is a valid Q-J decomposition, iff $\mathcal{Q}_0 \subseteq \mathcal{Q}$, and:

- P1. Every cluster $J \in \mathcal{J}$ is a small cluster with the bandwidth property, and every cluster $Q \in \mathcal{Q}$ is a critical cluster.
- P2. There is a set $N = \left\{\tau_e \mid e \in E^J\right\}$ of paths, called *tendrils*, where path τ_e connects e to some edge in E^Q, each edge in E^Q is an endpoint of at most one tendril, and the total congestion caused by N is at most 3. Moreover, the tendrils do not use edges $e = (u, v)$ where both u and v belong to clusters in $\mathcal{Q}$.
- P3. If $(S, \overline{S})$ is any cut in graph G, which is non-trivial and canonical w.r.t. $\mathcal{Q}$, then it is a large cut.

We refer to the clusters in $\mathcal{Q}$ as the Q-clusters, and to the clusters in $\mathcal{J}$ as the J-clusters. We note that Andrews [And10] has (implicitly) defined the Q-J decomposition, and suggested an algorithm for constructing it, by using the graph decomposition of Räcke [Räc02] as a black-box. The Räcke decomposition however gives very strong properties - stronger than one needs to construct a Q-J decomposition. We obtain a Q-J decomposition with slightly stronger properties by performing the decomposition directly, instead of using the Räcke's result as a black-box. For example, the tendrils in N only cause a constant congestion in our decomposition, instead of a logarithmic one, the well-linkedness of the J-clusters is $\mathrm{poly}\log\log n$ instead of $\mathrm{poly}\log n$, and we obtain a better relationship between the parameter L and the size of the minimum cut separating the Q-clusters. The algorithm for finding a Q-J decomposition is summarized in the next theorem, and its proof appears in the Appendix.

Theorem 10 *There is an efficient algorithm, that, given any graph G, and a set $\mathcal{Q}_0$ of disjoint critical clusters, such that any non-trivial canonical cut w.r.t. $\mathcal{Q}_0$ in G is large, produces a valid Q-J decomposition of G.*

Given a valid Q-J decomposition, it is not hard to construct a graph H with the desired properties. The following theorem mostly follows the construction of [And10], with some minor changes. We defer its proof to the Appendix.

Theorem 11 *Suppose we are given a valid Q-J decomposition $(\mathcal{Q}, \mathcal{J})$ for graph G. Then there is an efficient randomized algorithm to construct a graph H with $V(H) = \{v_C \mid C \in \mathcal{Q}\}$, and for each edge $e = (v_{C_1}, v_{C_2}) \in E(H)$, define a path P_e in graph G, connecting some vertex of C_1 to some vertex of C_2, such that for some value $\alpha^* = O(\log^8 n \,\mathrm{poly}\log\log n)$, the following properties hold w.h.p. for graph H:*

C1. For every cut (A, B) in graph H, there is a cut (A', B') in graph G, such that for each $Q \in \mathcal{Q}$, if $v_Q \in A$ then $Q \subseteq A'$, and if $v_Q \in B$ then $Q \subseteq B$, and $|E_G(A', B')| \leq \alpha^* \cdot |E_H(A, B)|$.

C2. The value of the minimum cut in H is at least $\frac{L}{\alpha^*}$.

C3. The paths in set $\mathcal{P}_H^E = \{P_e \mid e \in E(H)\}$ cause a constant congestion in graph G.

C4. For each critical cluster $C \in \mathcal{Q}$, let $\mathcal{G}_C$ be the grouping of the edges of $\mathrm{out}(C)$ given by Theorem 9. Then for each group $U \in \mathcal{G}_C$, at most two paths in $\mathcal{P}_H^E$ contain an edge of U as their first or last edge.

Once we compute the graph H, we can split it into graphs $H_1, \ldots, H_x$, as follows. For each $1 \leq j \leq x$, the set of vertices $V(H_j) = V(H)$. The sets of edges $E_1, \ldots, E_x$ are constructed as follows. Each edge $e \in E(H)$ independently chooses an index $j \in \{1, \ldots, x\}$ uniformly at random. Edge e is then added to graph H_j, where j is the index chosen by e. We use the following theorem (a re-statement of Theorem 2.1 from [Kar99]).

Theorem 12 ([Kar99]) *Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be any n-vertex graph with minimum cut value C. Assume that we obtain a sub-graph $\mathbf{G}' = (\mathbf{V}, \mathbf{E}')$, by adding every edge $e \in \mathbf{E}$ with probability p to $\mathbf{E}'$, and assume further that $C \cdot p > 48 \ln n$. Then with probability at least $1 - O(1/n^2)$, for every partition (A, B) of $\mathbf{V}$, $|E_{\mathbf{G}'}(A, B)| \geq \frac{p|E_G(A,B)|}{2}$.*

Therefore, if we select L so that $\frac{L}{x\alpha^*} > 48 \ln n$, then we can perform the graph splitting as described above, and from Theorem 12, for each $1 \leq j \leq x$, for each partition (A, B) of $V(H)$, $|E_{H_j}(A, B)| \geq \frac{|E_H(A,B)|}{2x}$ w.h.p.

4 An Algorithm for basic-ICF

The goal of this section is to prove Theorem 1. We start by proving the theorem for the special case where all demands are uniform, and all edge capacities are unit. That is, we are given a subset $\mathcal{M} \subseteq \mathcal{T} \times \mathcal{T}$ of the terminal pairs, and for each pair $(t, t') \in \mathcal{M}$, $D(t, t') = D$, while all other demands are 0. We extend this algorithm to handle arbitrary demands and edge capacities in Section I of the Appendix. We set the parameter $L = \Theta(\log^{24} n \operatorname{poly}\log\log n)$, and we define its exact value later.

Assume first that in the input instance G, any cut separating the set $\mathcal{T}$ of terminals has value at least L. We show in the next theorem, that in this case we can find an integral solution of value $\lambda_{\text{OPT}}/\operatorname{poly}\log n$ with constant congestion to instance $(G, \mathcal{D})$. The main idea is to use Theorem 11 to construct a graph H, where the initial set $\mathcal{Q}_0$ of critical clusters is $\mathcal{Q}_0 = \{\{t\} \mid t \in \mathcal{T}\}$. We then split H into $\operatorname{poly}\log n$ sub-graphs using the procedure outlined in Section 3, and use the algorithm of Rao and Zhou [RZ10] to route a polylogarithmic fraction of the demand pairs in each resulting subgraph. The proof of the next theorem is deferred to Appendix.

Theorem 13 *Let G be an instance of basic-ICF with unit edge capacities and a set $\mathcal{D}$ of uniform demands over the set $\mathcal{T}$ of terminals. Assume that every cut separating the terminals in $\mathcal{T}$ has size at least $L = \Theta(\log^{24} n \operatorname{poly}\log\log n)$. Then there is an efficient randomized algorithm that w.h.p. finds an integral solution of value λ_{OPT}/β with constant congestion, where $\beta = O(\log^{30} n \operatorname{poly}\log\log n)$ and λ_{OPT} is the value of the optimal fractional solution.*

In general, graph G may contain small cuts that separate its terminals. We get around this problem as follows. For each subset $S \subseteq V$ of vertices, let $\mathcal{T}_S = S \cap \mathcal{T}$ be the subset of terminals contained in S. We say that S is a *good subset* iff (1) Any cut in graph $G[S]$ separating the terminals in $\mathcal{T}_S$ has value at least L; and (2) $|\mathrm{out}(S)| \leq L \log k$. We first show that we can efficiently compute a good set S of vertices in graph G. We then decompose the set $\mathcal{D}$ of demands into two subsets: $\mathcal{D}_S$ containing demands for pairs in $\mathcal{T}_S$, and $\mathcal{D}'$ containing demands for all other pairs. Next, we apply Theorem 13 to instance $(G[S], \mathcal{D}_S)$, obtaining a collection $\mathcal{P}'$ of paths, and solve the problem recursively on instance $(G, \mathcal{D}')$, obtaining a collection $\mathcal{P}''$ of paths. Our final step is to carefully combine the two sets of paths to obtain the final solution $\mathcal{P}$. We start with the following lemma, whose proof is deferred to the Appendix, that allows us to find a good subset S of vertices efficiently.

Lemma 2 *Let $(G, \mathcal{D})$ be a basic-ICF instance with uniform demands and unit edge capacities, and a set $\mathcal{T}$ of terminals, where $|\mathcal{T}| \leq k$. Then there is an efficient algorithm that either finds a good set $S \subseteq V(G)$ of vertices, or establishes that every cut (A, B) separating the terminals of $\mathcal{T}$ in G has value $|E_G(A, B)| \geq L$.*

We use the following theorem, whose proof appears in the Appendix, to combine the solutions to the two sub-instances. In this theorem, we assume that we are given a good vertex set S, $\mathcal{T}_S = \mathcal{T} \cap S$, and $\mathcal{M}_S \subseteq \mathcal{M}$ is the subset of the demand pairs contained in S. We assume w.l.o.g. that $\mathcal{M}_S = \{(s_1, t_1), \ldots, (s_{k'}, t_{k'})\}$.

Theorem 14 *Suppose we are given a good vertex set S, and $\mathcal{M}_S \subseteq \mathcal{M}$ as above. Assume further that for each $1 \leq i \leq k'$, we are given a set $\mathcal{P}_i$ of N paths connecting s_i to t_i, such that all paths in set $\mathcal{P}' = \bigcup_{i=1}^{k'} \mathcal{P}_i$ are contained in $G[S]$, and set $\mathcal{P}'$ causes congestion at most γ in $G[S]$. Let $\mathcal{P}''$ be any set of paths in graph G, where each path in $\mathcal{P}''$ connects some pair $(s,t) \in \mathcal{M} \setminus \mathcal{M}_S$, and the congestion caused by the paths in $\mathcal{P}''$ is at most γ. Then we can efficiently find, for each $1 \leq i \leq k'$, a subset $\mathcal{P}_i^* \subseteq \mathcal{P}_i$ of at least $N - 2L\gamma \log n$ paths, and for each $P \in \mathcal{P}''$, find a path $\tilde{P}$ connecting the same pair of vertices as P, such that the total congestion caused by the set $\left(\bigcup_{i=1}^{k'} \mathcal{P}_i^*\right) \cup \left\{\tilde{P} \mid P \in \mathcal{P}''\right\}$ of paths is at most γ in graph G.*

We denote this algorithm by $\textsc{Reroute}(\mathcal{P}', \mathcal{P}'')$, and its output is denoted by $\tilde{\mathcal{P}}' = \bigcup_{i=1}^{k'} \mathcal{P}_i^*$, and $\tilde{\mathcal{P}}'' = \left\{\tilde{P} \mid P \in \mathcal{P}''\right\}$. We now complete the description of our algorithm, that we call RecursiveRouting.

We assume that we are given a graph G, a set $\mathcal{T}$ of at most k terminals, a set $\mathcal{M}$ of demand pairs, and real number $D > 0$, such that for each pair $(t, t') \in \mathcal{M}$, we can send $\lambda_{\text{OPT}} \cdot D$ flow units simultaneously in G with no congestion. If no cut of size less than L separates the terminals of $\mathcal{T}$, then we use Theorem 13 to find a set $\mathcal{P}$ of paths and return $\mathcal{P}$. Otherwise, we find a good vertex set $S \subseteq V(G)$ using Lemma 2. Let $\mathcal{T}_S = \mathcal{T} \cap S$, $\mathcal{M}_S \subseteq \mathcal{M}$ the subset of pairs contained in $\mathcal{T}_S$, $\mathcal{M}' = \mathcal{M} \setminus \mathcal{M}_S$. We then apply Theorem 13 to $(G[S], \mathcal{M}_S, D)$ to obtain a collection $\mathcal{P}'$ of paths, and invoke RecursiveRouting on $(G, \mathcal{M}', D)$ to obtain a set $\mathcal{P}''$ of paths. Finally, we apply procedure Reroute to sets $\mathcal{P}', \mathcal{P}''$ of paths to obtain the collections $\tilde{\mathcal{P}}', \tilde{\mathcal{P}}''$ of paths, and return $\mathcal{P} = \tilde{\mathcal{P}}' \cup \tilde{\mathcal{P}}''$.

Let β be the parameter from Theorem 13, and let γ be the congestion it guarantees. We can assume w.l.o.g. that $\frac{\lambda_{\text{OPT}}}{\beta} D \geq 4L\gamma \log n$, since otherwise $\lfloor \frac{\lambda_{\text{OPT}}}{4\beta L\gamma \log n} D \rfloor = 0$, and $\mathcal{P} = \emptyset$ is a good solution to the problem. We now prove that procedure RecursiveRouting produces a solution of value $\frac{\lambda_{\text{OPT}}}{4\beta}$ and congestion at most γ. The proof of the next lemma is deferred to the Appendix.

Lemma 3 *Let $\mathcal{P}$ be the output of procedure* RecursiveRouting. *Then for every pair $(s,t) \in \mathcal{M}$, at least $\lfloor \frac{\lambda_{OPT}}{4\beta} D \rfloor$ paths connect s to t in $\mathcal{P}$, and the paths in $\mathcal{P}$ cause congestion at most γ in G.*

This completes the proof of Theorem 1 for uniform demands and unit edge capacities. In Section I of the Appendix we extend this algorithm to arbitrary edge capacities and demands using standard techniques.

5 Hardness of basic-ICF

In this section we prove Theorem 2, by performing a simple reduction from the Congestion Minimization problem. We use the following theorem, due to Andrews and Zhang [AZ07].

Theorem 15 *Let G be any n-vertex graph with unit edge capacities, and let $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ be a collection of source-sink pairs. Then, unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\text{poly} \log n})$, no efficient algorithm can distinguish between the following two cases:*

- *(*Yes-Instance*): There is a collection $\mathcal{P}$ of paths that causes congestion 1, and for each $1 \leq i \leq k$, there is a path connecting s_i to t_i in $\mathcal{P}$.*
- *(*No-Instance*): For any collection $\mathcal{P}$ of paths that contains, for each $1 \leq i \leq k$, a path connecting s_i to t_i, the congestion due to $\mathcal{P}$ is at least $\Omega(\log \log n / \log \log \log n)$.*

Let G be any n-vertex graph with unit edge capacities and a set $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of source-sink pairs. We build a new graph G', where we replace every edge in G by D parallel edges. Observe that if G is a Yes-Instance, then there is a collection $\mathcal{P}$ of paths that causes congestion 1 in G, and every demand pair $(s_i, t_i) \in \mathcal{M}$ is connected by a path in $\mathcal{P}$. We then immediately obtain a collection $\mathcal{P}'$ of paths in graph G' that causes congestion 1, and every demand pair (s_i, t_i) is connected by D paths, by simply taking D copies of each path in $\mathcal{P}$. Assume now that G is a No-Instance, and assume for contradiction that there is a collection $\mathcal{P}'$ of paths in graph G' that causes congestion at most c, and every demand pair (s_i, t_i) is connected by at least

one path in $\mathcal{P}'$. Then set $\mathcal{P}'$ of paths defines a collection $\mathcal{P}$ of paths in graph G, that causes congestion at most Dc, and every demand pair (s_i, t_i) is connected by at least one path in $\mathcal{P}$.

6 Hardness of group-ICF

The goal of this section is to prove Theorem 3. We start by introducing a new scheduling problem, called Max-Min Interval Scheduling (MMIS), and show that it can be cast as a special case of group-ICF. We then show that MMIS is hard to approximate.

In the MMIS problem, we are given a collection $\mathcal{J} = \{1, \ldots, n\}$ of n jobs, and for each job j, we are given a set $\mathcal{I}_j$ of **disjoint** closed intervals on the time line. Given any subset $\mathcal{I}$ of intervals, the *congestion* of $\mathcal{I}$ is the maximum, over all time points t, of the number of intervals in $\mathcal{I}$ containing t. Speaking in terms of scheduling, this is the number of machines needed to schedule the jobs during the time intervals in $\mathcal{I}$. The goal of MMIS is to find, for each job j, a subset $\mathcal{I}^*_j \subseteq \mathcal{I}_j$ of intervals such that, if we denote $\mathcal{I}^* = \bigcup_{j=1}^{n} \mathcal{I}^*_j$, then the intervals in $\mathcal{I}^*$ are disjoint (or equivalently cause congestion 1). The value of the solution is the minimum, over all jobs $j \in \mathcal{J}$, of $|\mathcal{I}^*_j|$. Given an instance $\mathcal{J}$ of MMIS, we denote by $N(\mathcal{J})$ the total number of intervals in $\bigcup_{j \in \mathcal{J}} \mathcal{I}_j$. In the following theorem, whose proof appears in Appendix, we relate MMIS to group-ICF.

Theorem 16 *Suppose we are given any instance $\mathcal{J}$ of* MMIS*. Then we can eficiently construct an instance $(G, \mathcal{D})$ of* group-ICF *on a line graph, such that $|V(G)| \leq 2N(\mathcal{J})$ and the following holds:*

- *If* OPT *is the value of the optimal solution to $\mathcal{J}$ with congestion* 1*, then there is a collection $\mathcal{P}^*$ of edge-disjoint paths in G, where each pair (S_i, T_i) is connected by* OPT *paths in $\mathcal{P}^*$, and*
- *Given any solution $\mathcal{P}^*$ in which every pair (S_i, T_i) is connected by at least D' paths, and the total congestion cased by $\mathcal{P}^*$ is bounded by c, we can efficiently find a solution $\mathcal{I}^*$ to instance $\mathcal{J}$ of value D' and congestion c.*

We note that a scheduling problem closely related to MMIS is Machine Minimization Job Scheduling, where the goal is to select one time interval for every job, so as to minimize the total congestion. This problem admits an $O(\log n / \log \log n)$-approximation via the Randomized LP-Rounding technique of Raghavan and Thompson [RT87], and Chuzhoy and Naor [CN06] have shown that it is $\Omega(\log \log n)$-hard to approximate. In the following theorem, we prove hardness of the MMIS problem, which, combined with Theorem 16, immediately implies Theorem 3. Its proof very closely follows the hardness of approximation proof of [CN06] for Machine Minimization Job Scheduling and is deferred to the Appendix.

Theorem 17 *Suppose we are given an instance $\mathcal{J}$ of* MMIS*, and let $N = N(\mathcal{J})$. Let c be any integer, $0 < c \leq O(\log \log N)$ and let $D = O\left(N^{1/2^{2c+3}}\right)$. Then unless* NP $\subseteq$ DTIME$(n^{O(\log \log n)})$*, no efficient algorithm can distinguish between the following two cases:*

- *(*YES-INSTANCE*): the value of the optimal solution with congestion* 1 *is at least D, and*
- *(*NO-INSTANCE*): any solution $\mathcal{I}^*$, where for all $j \in J$, $|\mathcal{I}^*_j| \geq 1$, causes congestion at least c.*

7 An Algorithm for group-ICF

Due to lack of space, we only provide a high level overview of the algorithm here, and provide the details in Section H of the Appendix. We first assume that all edge capacities are unit and all demands are uniform, and then extend our results to general demands and edge capacities. By using the standard randomized rounding technique, we can assume that the fractional solution is $1/m$-integral, where $m = O(\log n)$. That is, we obtain a set $\mathcal{P} = \left\{P^i_j \mid 1 \leq i \leq k, 1 \leq j \leq mD\right\}$ of paths that cause congestion $2m$ in G, and for each $1 \leq i \leq k$, each path P^i_j for $1 \leq j \leq mD$ connects a vertex of S_i to a vertex of T_i. Let $\mathcal{M}$ denote the pairs of endpoints of the paths in $\mathcal{P}$. We then define two special cases of group-ICF. The first special case is a split instance, and is defined as follows. We are given a partition $\mathcal{C}$ of $V(G)$, such that every path in $\mathcal{P}$ is contained in some cluster of $\mathcal{C}$, and moreover for each $1 \leq i \leq k$, the number of paths connecting the vertices of S_i to the vertices of T_i in $\mathcal{P}$ that are contained in any single cluster $C \in \mathcal{C}$ is at most $Dm/\operatorname{poly}\log n$. For

each cluster $C \in \mathcal{C}$, let $\mathcal{M}_C \subseteq \mathcal{M}$ be the subset of pairs whose corresponding paths in $\mathcal{P}$ are contained in C. We then apply the algorithm of [Chu11] to each cluster $C \in \mathcal{C}$ separately polylogarithmic number of times. In each such iteration, we find a collection of paths connecting a $1/\mathrm{polylog(n)}$-fraction of pairs in $\mathcal{M}_C$. We then select one of the resulting solutions independently at random. Since for each pair (S_i, T_i), every cluster only contains a small fraction of paths connecting S_i to T_i, we are guaranteed that w.h.p. every pair is connected by $D/\operatorname{poly}\log n$ paths in the final solution. The second special case is when we can find a good Q-J decomposition for the group-ICF instance. A Q-J decomposition is good iff no pair $(s,t) \in \mathcal{M}$ is contained in any cluster $X \in \mathcal{J} \cup \left(\bigcup_{Q \in \mathcal{Q}} \pi(Q)\right)$ (here, $\pi(Q)$ is the partition of the critical cluster Q into small clusters). We show an algorithm to find a $\operatorname{poly}\log n$-approximate solution with constant congestion for instances where a good Q-J decomposition is given. The algorithm is similar to the algorithm for the basic-ICF problem. Finally, given any group-ICF instance, we decompose it into several sub-instances, such that every edge of G only belongs to a constant number of such sub-instances, and each sub-instance is either a split instance, or we can find a good Q-J decomposition for it.

Acknowledgements. The second author would like to thank Matthew Andrews for sharing an early version of his paper and for many interesting discussions. She also thanks Sanjeev Khanna for many interesting discussions about routing problems. The third author thanks Chandra Chekuri for useful discussions.

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.

[ALM+98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[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.

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

[AS98] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[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.

[CHR03] M. Conforti, R. Hassin, and R. Ravi. Reconstructing flow paths. *Operations Research Letters*, 31:273–276, 2003.

[Chu11] Julia Chuzhoy. Routing in undirected graphs with constant congestion. *arXiv:1107.2554v1.*, Full version at http://ttic.uchicago.edu/~cjulia/publications.html, 2011.

[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.

[CKS06] 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.

[CN06] Julia Chuzhoy and Joseph (Seffi) Naor. New hardness results for congestion minimization and machine scheduling. *J. ACM*, 53(5):707–721, 2006.

[GS62] David Gale and Lloyd Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 1:9–14, 1962.

[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.

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

[Kar93] David R. Karger. Global min-cuts in rnc, and other ramifications of a simple min-out algorithm. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, SODA '93, pages 21–30, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[Kar99] David R. Karger. Random sampling in cut, flow, and network design problems. *Mathematics of Operations Research*, 24:383–413, 1999.

[Kar00] David R. Karger. Minimum cuts in near-linear time. *J. ACM*, 47:46–76, January 2000.

[KS96] David R. Karger and Clifford Stein. A new approach to the minimum cut problem. *J. ACM*, 43:601–640, July 1996.

[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.

[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.

[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.

[Raz98] R. Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, 1998.

[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.

A List of Parameters

$\alpha_{\text{ARV}}(n)$	$O(\sqrt{\log n})$	Approximation factor of algorithm $\mathcal{A}_{\text{ARV}}$ for sparsest cut
$\beta_{\text{FCG}}(k)$	$O(\log k)$	Flow-cut gap for undirected graphs
L	$\text{poly}\log n$	Threshold for defining small and large clusters
α_W	$\Omega(1/\log^{1.5} n)$	Well-linkedness for large clusters given by Theorem 6
α_S	$\Omega\left(1/(\log\log n)^{1.5}\right)$	Parameter for bandwidth condition of small clusters
α_{BW}	$\alpha_W \cdot \alpha_S = \Omega\left(1/(\log n \log\log n)^{1.5}\right)$	Parameter for bandwidth condition of large clusters
λ	$\frac{\alpha_{\text{BW}}}{8\alpha_{\text{ARV}}(n)} = \Omega(\frac{1}{\log^2 n \cdot (\log\log n)^{1.5}})$	Parameter for weight condition
η^*	$\frac{2\beta_{\text{FCG}}(L)}{\alpha_S} = O((\log\log n)^{2.5})$	A large cut contains more than $L/(4\eta^*)$ edges
α_{EDP}	$\text{poly}\log k$	Approximation factor for the algorithm for EDP from Theorem 5
η_{EDP}	14	Congestion of the algorithm for EDP from Theorem 5
α^*	$O(\log^8 n \,\text{poly}\log\log n)$	A factor up to which the cuts are preserved in graph H in Theorem 11
α_{RZ}	$O(\log^{10} n)$	Approximation factor from the algorithm of [RZ10] for EDP
L_{RZ}	$\Omega(\log^5 n)$	Requirement on the size of minimum global cut in the algorithm of [RZ10]

Additional parameters for the algorithm for group-ICF.

L	$O(\log^{25} n)$	Threshold for defining small and large clusters
m	$O(\log n)$	Parameter for canonical instances
Δ	$O(k\,\text{poly}\log n)$	We require that $D \geq 640\Delta m\alpha_{\text{EDP}}\log^2 n = k\,\text{poly}\log n$

B Gaps Between Fractional and Integral group-ICF

We start by showing that if no congestion is allowed, then the ratio between λ_{OPT} - the optimal fractional solution and λ^* - the optimal integral solution can be as large as $\Omega(\sqrt{n})$ for group-ICF, even if $k = 2$.

Our gap example is a slight modification of the $n \times n$ grid. We start from a grid G with $V(G) = \{v(i,j)|(i,j) \in [n] \times [n]\}$, where $v(i,j)$ denotes the vertex on the ith row and jth column. We add new sets $S_1 = \{s_1, \ldots, s_n\}$, $T_1 = \{t_1, \ldots, t_n\}$, $S_2 = \{s'_1, \ldots, s'_n\}$ and $T_2 = \{t'_1, \ldots, t'_n\}$ of vertices (see Figure 1). For each $1 \leq j \leq n$, we connect s_j to $v(j,1)$, $v(j,n)$ to t_j, $v(1,j)$ to s'_j, and $v(n,j)$ to t'_j. Finally, for each $i,j \in [n]$, we replace vertex $v(i,j)$ by a gadget that is shown in the figure. Denote the resulting graph by G'.

The fractional solution can send $n/2$ flow units from S_i to T_i for $i \in \{1,2\}$ in G', where for each j, s_j sends a $\frac{1}{2}$-flow unit to t_j using the paths corresponding to the horizontal grid lines, and s'_j sends a $\frac{1}{2}$-flow unit to

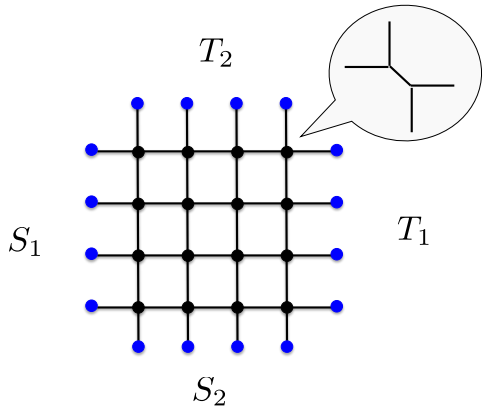

Figure 1: The gap example construction

t'_j, using the paths corresponding to the vertical grid lines. It is easy to see that the value of the integral solution is 0: Assume that an integral solution $\mathcal{P}$ contains a path P_1 connecting a vertex of S_1 to a vertex of T_1, and a path P_2 connecting a vertex of S_2 to a vertex of T_2. Consider the corresponding paths P'_1, P'_2 in the $n \times n$ grid. Then paths P'_1 and P'_2 must cross at some vertex $v(i,j)$ of the grid. But then P_1 and P_2 must share an edge from the gadget corresponding to $v(i,j)$. Since the number of vertices in G' is $N = O(n^2)$, this shows a gap of $\Omega(\sqrt{N})$ between the optimal fractional and the optimal integral solutions.

Next, we show a gap example when congestion is allowed. Let c be a constant and D be a parameter. We show an instance where we can fractionally send $D = \Theta(n^{1/(2c)})$ flow units for every group, while any integral solution containing at least 1 path for every group will cause congestion $2c$.

Our construction is iterative. An input to iteration j is a partition $\mathcal{I}_{j-1}$ of the interval $[0,1]$ of the real line into $(2Dc)^{j-1}$ sub-intervals, that we call *level-*$(j-1)$ intervals. At the beginning, $\mathcal{I}_0$ consists of a single level-0 interval $[0,1]$. An iteration is executed as follows. Let $I \in \mathcal{I}_{j-1}$ be any level-$(j-1)$ interval, and let ℓ, r be its left and right endpoints, respectively. We partition I into $2cD$ equal-length segments by $2cD+1$ new terminals $s_1(I) = \ell, t_1(I), s_2(I), t_2(I), \cdots, s_{cD}(I), t_{cD}(I), s_{cD+1}(I) = r$. We then define a new demand pair $S(I), T(I)$, where $S(I) = \{s_h(I)\}_{h=1}^{cD+1}$, $T(I) = \{t_h(I)\}_{h=1}^{cD}$. Each of the new sub-intervals of I becomes a level-j interval, and serves as the input to the next iteration.

The final construction is the line graph G obtained after $2c$ iterations. Notice that $|V(G)| = (2cD)^{2c} + 1$, and the number of the demand pairs is $k = \sum_{j=0}^{2c-1}(2cD)^j = \Theta((2cD)^{2c-1})$.

In the fractional solution, for each $1 \leq j \leq 2c$, for each level-$(j-1)$ interval I, each sink terminal $t_h(I)$ receives $1/(2c)$ flow units from each of the two source terminals to its right and to its left. Thus, we send $\frac{1}{2c} \cdot 2cD = D$ flow units between the vertices of $S(I)$ and the vertices of $T(I)$. The congestion of this fractional solution is 1, since each edge is used by $2c$ groups, with each group using $1/(2c)$ of its capacity.

Consider now any integral solution that contains at least one path connecting every demand pair $(S(I), T(I))$. We show that the congestion of this solution is $2c$. Let I_0 be the unique level-0 interval $[0,1]$, and consider the path P_1 connecting some terminal in $S(I_0)$ to some terminal in $T(I_0)$. Then path P must contain some level-1 interval I_1. We then consider the path P_2 connecting some terminal of $S(I_1)$ to some terminal of $T(I_1)$. This path in turn must contain some level-2 interval I_2. We continue like this until we reach level $(2c)$, thus constructing a collection $I_1, \ldots, I_{2c}$ of nested intervals. Each such interval is contained in a distinct path that the solution uses. Therefore, the congestion of the solution is at least $2c$.

C Proof of Theorem 6

The main difference of this well-linked decomposition from the standard one is that it is "one-shot": given a current set S of vertices, we iteratively find a subset $R \subseteq S$, where R is well-linked itself, add R to the partition $\mathcal{W}$, update S to be $S \setminus R$, and then continue with the updated set S. Unlike the standard well-linked

decompositions, we do not recurse inside the set R, which is guaranteed to be well-linked. We will still be able to bound the number of partition edges $\sum_{R\in\mathcal{W}} |\operatorname{out}(R)|$ in terms of $|\operatorname{out}(S)|$ as before, and moreover construct the tendrils connecting every edge in $\bigcup_{R\in\mathcal{W}} \operatorname{out}(R)$ to the edges of $\operatorname{out}(S)$ with constant congestion.

We start with some definitions. We assume that we are given a graph G, and a subset $S \subseteq V(G)$, with $|\operatorname{out}(S)| = k'$. Given any integer r, let $\alpha(r) = \frac{1}{10}\left(1 - \frac{1}{\log k'}\right)^{\log r}$. Notice that for $1 \leq r \leq k'$, $\frac{1}{20e} \leq \alpha(r) \leq \frac{1}{10}$.

Definition 3 *Let R be any subset of vertices of S, and let (X, Y) be any partition of R, with $T_X = \operatorname{out}(R) \cap \operatorname{out}(X)$, $T_Y = \operatorname{out}(R) \cap \operatorname{out}(Y)$, and $|T_X| \leq |T_Y|$. We say that X is a* sparse cut *for R, if*

$$|E_G(X, Y)| < \alpha(r)|T_X|,$$

where $r = |T_X|$.

Observe that if X is a sparse cut, and $Y = S \setminus X$, then $|\operatorname{out}(X)|, |\operatorname{out}(Y)| \leq |\operatorname{out}(S)|$. Notice also that since $\alpha(r) \geq \frac{1}{20e}$, if set S has no sparse cuts, then it is $\frac{1}{20e}$-well-linked.

We now proceed as follows. First, we obtain a non-constructive well-linked decomposition, that gives slightly weaker guarantees. In particular, it does not ensure that the small clusters are α_S-well-linked. We then turn this decomposition into a constructive one, while slightly weakening its parameters. Finally, we show an algorithm, that improves the parameters of the decomposition, by recursively applying it to some of the clusters.

C.1 Non-constructive decomposition

We show an algorithm to construct the decomposition, whose running time is exponential in k'. We later turn this algorithm into an efficient one, with slightly worse parameters. We note that in the current section we do not ensure that every small cluster is α_S-well linked, but we obtain this guarantee in our final decomposition. The decomposition algorithm works as follows.

- Start with $\mathcal{W} = \emptyset$, $S' = S$.
- While S' contains any sparse cut:
  - Let X be a sparse cut minimizing $|T_X|$; if there are several such cuts, choose one minimizing the number of vertices $|X|$.
  - Add X to $\mathcal{W}$ and set $S' := S' \setminus X$.
- Add S' to $\mathcal{W}$

We now proceed to analyze the algorithm. First, we show that all sets added to $\mathcal{W}$ are well-linked, in the next lemma.

Lemma 4 *Every set added to $\mathcal{W}$ is $\frac{1}{80e\cdot\log k'}$-well linked.*

Proof: First, the last set added to $\mathcal{W}$ must be $\frac{1}{20e}$-well-linked, since S' does not contain any sparse cuts at this point.

Consider now some iteration of the algorithm. Let S' be the current set, X the sparse cut we have selected, and $Y = S' \setminus X$. We denote $T_X = \operatorname{out}(X) \cap \operatorname{out}(S')$, $T_Y = \operatorname{out}(Y) \cap \operatorname{out}(S')$, $\Gamma = E_G(X, Y)$, $r = |T_X|$, and recall that $|T_X| \leq |T_Y|$, and $|\Gamma| < r \cdot \alpha(r)$.

Assume for contradiction that X is not $\frac{1}{80e\cdot\log k'}$-well linked, and let (A, B) be a violating cut. We can assume that A and B are not sparse cuts for S': otherwise, we should have added A or B to $\mathcal{W}$ instead of X. The following claim will then finish the proof, as it contradicts the fact that (A, B) is a violating cut.

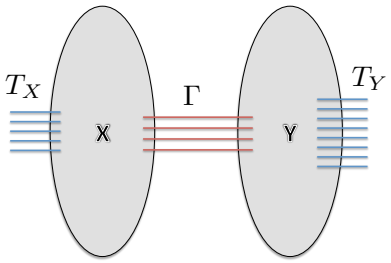

Figure 2: Illustration for Lemma 4

Claim 1 *If X is a sparse cut for S', (A,B) is a partition of X, and A,B are not sparse cuts for S', then*

$$|E(A,B)| \geq \frac{1}{80e \cdot \log k'} \min\left\{|\operatorname{out}(A) \cap \operatorname{out}(X)|, |\operatorname{out}(B) \cap \operatorname{out}(X)|\right\}.$$

Proof:

We denote $T'_X = T_X \cap \operatorname{out}(A)$, $T''_X = T_X \cap \operatorname{out}(B)$, $\Gamma' = \Gamma \cap \operatorname{out}(A)$, $\Gamma'' = \Gamma \cap \operatorname{out}(B)$, and we assume w.l.o.g. that $|T'_X| \leq |T''_X|$ (notice that it is possible that $T'_X = \emptyset$). Let $E' = E(A,B)$, $|T'_X| = r'$, and $|T''_X| = r''$ (see Figure 3).

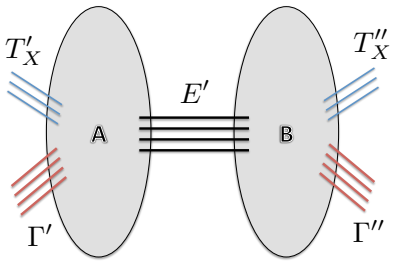

Figure 3: Illustration for Claim 1

Since B is not sparse for S', $|E'| + |\Gamma''| \geq r'' \cdot \alpha(r'')$, and equivalently, $|\Gamma''| \geq r'' \cdot \alpha(r'') - |E'| \geq r''\alpha(r) - |E'|$ (since $r'' \leq r$ and so $\alpha(r'') \geq \alpha(r)$).

On the other hand, since X is a sparse cut, $|\Gamma'| + |\Gamma''| < r \cdot \alpha(r)$, so $|\Gamma'| < r'\alpha(r) + |E'|$. Notice that if $T'_X = \emptyset$, then $|\Gamma'| < |E'|$ must hold, contradicting the assumption that (A,B) is a violating cut. Therefore, we assume from now on that $T'_X \neq \emptyset$.

Recall that $r' \leq r/2$, and so $\log r' \leq \log r - 1$. Therefore,

$$\alpha(r') = \frac{1}{10}\left(1 - \frac{1}{\log k'}\right)^{\log r'} \geq \frac{1}{10}\left(1 - \frac{1}{\log k'}\right)^{\log r - 1} = \alpha(r) / \left(1 - \frac{1}{\log k'}\right)$$

So $\alpha(r) \leq \left(1 - \frac{1}{\log k'}\right) \cdot \alpha(r')$. We therefore get that:

$$|\Gamma'| < r'\alpha(r') \cdot \left(1 - \frac{1}{\log k'}\right) + |E'| \tag{1}$$

Since A is not a sparse cut for S', $|E'| + |\Gamma'| \geq r' \cdot \alpha(r')$, and $|\Gamma'| \geq r' \cdot \alpha(r') - |E'|$.

Combining this with Equation (1), we get that

$$2|E'| + r'\alpha(r') \cdot \left(1 - \frac{1}{\log k'}\right) > r' \cdot \alpha(r'),$$

and rearranging the sides:

$$|E'| > \frac{r'\alpha(r')}{2\log k'} \tag{2}$$

Substituting this in Equation (1), we get that:

$$|\Gamma'| < r'\alpha(r') \cdot \left(1 - \frac{1}{\log k'}\right) + |E'| < r'\alpha(r') + |E'| < 3|E'|\log k'$$

and so

$$|E'| > \frac{|\Gamma'|}{3\log k'} \tag{3}$$

Combining Equations (2) and (3), we get that

$$|E'| > \frac{|\Gamma'|}{6\log k'} + \frac{r'\alpha(r')}{4\log k'} > \frac{|\Gamma'| + r'}{80e \cdot \log k'}.$$

We conclude that

$$|E(A,B)| \geq \frac{1}{80e \cdot \log k'} |\operatorname{out}(A) \cap \operatorname{out}(X)|.$$

□ □

Constructing the tendrils The next lemma will be useful in constructing the tendrils.

Lemma 5 *Let X be any sparse cut added to $\mathcal{W}$ during the execution of the algorithm, $T_X = \operatorname{out}(X) \cap \operatorname{out}(S')$, where S' is the current set, and $\Gamma = E(X, S' \setminus X)$. Then there is a flow F in $G[X]$ with the following properties:*

- *Each edge in Γ sends one flow unit.*
- *Each edge in T_X receives $1/10$ flow unit.*
- *Congestion at most 1.*

Proof: We set up the following single source-sink network. Start with graph G, and for each edge $e \in \operatorname{out}(X)$, subdivide e by a vertex t_e. Let H be the sub-graph of the resulting graph induced by $X \cup \{t_e \mid e \in \operatorname{out}_G(X)\}$. Unify all vertices t_e for $e \in T_X$ into a source s, and unify all vertices t_e with $e \in \Gamma$ into a sink t. The capacities of all edges are unit, except for the edges adjacent to the source, whose capacities are $1/10$. Let H' denote the resulting flow network. The existence of the required flow in $G[X]$ is equivalent to the existence of an s-t flow of value $|\Gamma|$ in H'.

Assume for contradiction that such flow does not exist. Then there is a cut (A, B) in H' with $s \in A$, $t \in B$, and $|E(A,B)| < |\Gamma|$. Let $T_1 = E(A,B) \cap T_X$, $T_2 = T_X \setminus T_1$, and similarly, $\Gamma_1 = E(A,B) \cap \Gamma$, $\Gamma_2 = \Gamma \setminus \Gamma_1$. Let $E' = E(A,B) \setminus (\Gamma_1 \cup T_1)$.

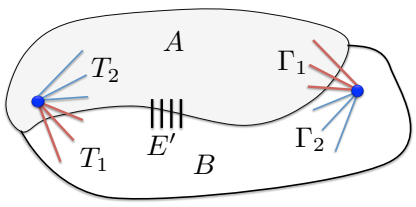

Figure 4: Illustration for Lemma 5

We then have that $|T_1|/10 + |\Gamma_1| + |E'| < |\Gamma|$. We claim that $(A \setminus \{s\})$ is a sparse cut for S, and so it should have been chosen instead of X. Indeed, since $|T_1|/10 < |\Gamma|$, and $|T_X| \geq 10|\Gamma|$, $T_2 \neq \emptyset$.

Let $|T_1| = r'$, $|T_2| = r''$. In order to prove that $(A \setminus \{s\})$ is a sparse cut, it is enough to prove that:

$$|E'| + |\Gamma_1| < r''\alpha(r'').$$

Recall that $|E'| + |\Gamma_1| < |\Gamma| - r'/10 < r\alpha(r) - r'/10$, since $|\Gamma| < r \cdot \alpha(r)$, because X is sparse for S'.

Since $\alpha(r) \leq \frac{1}{10}$, we get that $r'/10 \geq r'\alpha(r)$. We conclude that $|E'| + |\Gamma_1| < r\alpha(r) - r'\alpha(r) = r''\alpha(r) \leq r''\alpha(r'')$, a contradiction. □

Corollary 2 *Let $\mathcal{W}$ be the final partition of S produced by the algorithm, and let $E^* = \left(\bigcup_{R\in\mathcal{W}} \operatorname{out}(R)\right) \setminus \operatorname{out}(S)$ be the set of the new edges of the partition. Then there is a flow $F : E^* \rightsquigarrow \operatorname{out}(S)$ in graph $G[S] \cup \operatorname{out}(S)$, where each edge in E^* sends one flow unit, each edge in $\operatorname{out}(S)$ receives at most 0.2 flow units, and the total congestion is at most 1.2.*

Proof: For each edge $e \in E^* \cup \operatorname{out}(S)$, we will define a flow F_e, connecting e to the edges in $\operatorname{out}(S)$, and the final flow F will be the union of these flows over all edges $e \in E^*$. At the beginning, each edge $e \in \operatorname{out}(S)$ sends one flow unit to itself.

We now follow the decomposition algorithm, and maintain the following invariant: at each step of the algorithm, for each edge $e \in \left(\bigcup_{R\in\mathcal{W}} \operatorname{out}(R)\right)$, the flow f_e is already defined. In particular, if S' is the current set, then for each edge $e \in \operatorname{out}(S')$, the flow f_e is already defined. Consider a step of the algorithm, where we find a sparse cut X for S', and add X to $\mathcal{W}$. Let $\Gamma = E(X, S' \setminus X)$, and let $T_X = \operatorname{out}(X) \cap \operatorname{out}(S')$. From Lemma 5, there is a flow $F_1 : \Gamma \rightsquigarrow T_X$, where each edge $e \in \Gamma$ sends one flow unit, each edge in T_X receives at most 0.1 flow unit, the congestion is 1, and the flow is contained in $G[X]$. Consider some edge $e \in \Gamma$. Flow f_e is defined as follows: we use all flow-paths originating in e in F_1. For each such flow-path P, if $e' \in T_X$ is the other endpoint of P, then we concatenate P with $f_{e'}$, scaled down by factor $c(e') \leq 0.1$, where $c(e')$ is the total flow that edge e' receives in F_1.

This completes the description of the flow F. In order to bound the amount of flow that each edge $e \in \operatorname{out}(S)$ receives, notice that this flow forms a geometric progression: namely, if E_1 is the set of edges that send flow directly to e, E_2 is the set of edges that send flow directly to edges of E_1, and so on, then the total flow that e receives from E_1 is at most 0.1, the total flow that it receives from E_2 is at most 0.01 and so on. Therefore, the total flow that every edge in $\operatorname{out}(S)$ receives is bounded by 0.2. Congestion on edges is bounded by 1.2 similarly. □

The next corollary follows from Corollary 2 and the integrality of flow.

Corollary 3 *There is a collection $N = \{\tau_e \mid e \in E^*\}$ of paths (called tendrils) in graph $G[S]$, where each path τ_e connects e to some edge in $\operatorname{out}(S)$, the total congestion caused by path in N is at most 2, and each edge in S serves as an endpoint of at most one tendril.*

C.2 Constructive Version

In order to obtain an efficient algorithm for finding the decomposition, we use an approximation algorithm $\mathcal{A}_{\text{ARV}}$ for the Sparsest Cut problem. As before, we start with $\mathcal{W} = \emptyset$, and $S' = S$. We then perform a number of iterations, which are executed as follows. We set up an instance of the sparsest cut problem on graph $G[S']$, with the set $\operatorname{out}(S')$ of edges acting as terminals (imagine placing a terminal t_e on each edge $e \in \operatorname{out}(S')$). We then apply algorithm $\mathcal{A}_{\text{ARV}}$ to the resulting instance of the sparsest cut problem. Let (X, Y) be the output of algorithm $\mathcal{A}_{\text{ARV}}$, and assume w.l.o.g. that $|\operatorname{out}(X) \cap \operatorname{out}(S')| \leq |\operatorname{out}(Y) \cap \operatorname{out}(S')|$. If $|E(X, Y)| \geq \frac{1}{20e}|\operatorname{out}(A) \cap \operatorname{out}(S')|$, then we are guaranteed that set S' is $\frac{1}{20e\cdot\alpha_{\text{ARV}}(k')}$-well linked. In this case, we simply add S' to $\mathcal{W}$, and finish the algorithm.

Assume now that $|E(X, Y)| < \frac{1}{20e}|\operatorname{out}(X) \cap \operatorname{out}(S')|$. Clearly, this means that X is a sparse cut. Let $T_X = \operatorname{out}(X) \cap \operatorname{out}(S')$, and $\Gamma = E(X, Y)$. We say that a flow $F : \Gamma \rightsquigarrow T_X$ is a good flow iff every edge in

Γ sends one flow unit, every edge in T_X receives at most 0.1 flow units, the flow is contained in $G[X]$, and it causes congestion at most 1. If we are to add X to $\mathcal{W}$, we would like to ensure that X is well-linked, and a good flow F exists for X. In the next theorem we show that if this is not the case, then we can find another sparse cut $A \subseteq X$, such that either $|\operatorname{out}(A) \cap \operatorname{out}(S')| < |\operatorname{out}(X) \cap \operatorname{out}(S')|$, or $|\operatorname{out}(A) \cap \operatorname{out}(S')| = |\operatorname{out}(X) \cap \operatorname{out}(S')|$ and $|A| < |X|$.

Lemma 6 *Assume that X is a sparse cut for S'. Then either X is $\frac{1}{80e \cdot \log k' \cdot \alpha_{\text{ARV}}(k')}$-well-linked and has a good flow F, or we can efficiently find another sparse cut $A \subseteq X$, such that either $|\operatorname{out}(A) \cap \operatorname{out}(S')| < |\operatorname{out}(X) \cap \operatorname{out}(S')|$, or $|\operatorname{out}(A) \cap \operatorname{out}(S')| = |\operatorname{out}(X) \cap \operatorname{out}(S')|$ and $|A| < |X|$.*

We prove the lemma below, but we first complete the description of the algorithm. Let $\alpha_W(k') = \frac{1}{80e \cdot \log k' \alpha_{\text{ARV}}(k')}$. Once we find a sparse cut X, we apply Lemma 6 to X. If the outcome is that X is well-linked and has a good flow, then we simply add X to $\mathcal{W}$, set $S' := S' \setminus X$, and continue to the next iteration of the decomposition. Otherwise, if we have found a sparse cut $A \subseteq X$ as above, then we replace X by A, and repeat this step again. We continue applying Lemma 6 to the current set X, until we obtain a set that is $\alpha_W(k')$-well-linked and contains a good flow. We then add X to $\mathcal{W}$. Since at every step we are guaranteed that either $|T_X|$ decreases, or $|T_X|$ stays the same but $|X|$ decreases, we are guaranteed that after at most $k'n$ applications of Lemma 6 we will obtain a set X that can be added to $\mathcal{W}$. We now give a proof of Lemma 6.

Proof: [of Lemma 6] The proof consists of two steps. First, we try to find a good flow from Γ to T_X exactly as in Lemma 5. We build a flow network H' exactly as in the proof of Lemma 5. If such flow does not exist, then we obtain a cut (A, B) as before. From the proof of Lemma 6, in this case A is a sparse cut for S', and moreover either $|\operatorname{out}(A) \cap \operatorname{out}(S')| < |\operatorname{out}(X) \cap \operatorname{out}(S')|$, or $|\operatorname{out}(A) \cap \operatorname{out}(S')| = |\operatorname{out}(X) \cap \operatorname{out}(S')|$ and $|A| < |X|$. So if X does not contain a good flow, we can return a cut A as required. If X contains a good flow, then we perform the following step.

We run the algorithm $\mathcal{A}_{\text{ARV}}$ on the graph $G[X]$, where the edges of $\operatorname{out}(X)$ serve as terminals. Let (A, B) be the resulting cut. If $|E(A, B)| \geq \frac{1}{80e \cdot \log k'} \min\{|\operatorname{out}(A) \cap \operatorname{out}(X)|, |\operatorname{out}(B) \cap \operatorname{out}(X)|\}$, then we are guaranteed that X is $\frac{1}{80e \cdot \log k' \cdot \alpha_{\text{ARV}}(k')}$-well-linked as required. Otherwise, from the proof of Claim 1, either A or B must be a sparse cut, satisfying the conditions of the lemma. □

We summarize the algorithm, which we call from now on the *basic well-linked decomposition*, in the next theorem.

Theorem 18 (Basic well-linked decomposition) *There is an efficient algorithm, that, given a set S of vertices, with $|\operatorname{out}(S)| = k'$, produces a partition $\mathcal{W}$ of S with the following properties.*

- *For each set $R \in \mathcal{W}$, $|\operatorname{out}(R)| \leq k'$ and R is $\alpha_W(k') = \Omega\left(1/\log^{1.5} k'\right)$-well-linked.*
- *Let $E^* = \left(\bigcup_{R \in \mathcal{W}} \operatorname{out}(R)\right) \setminus \operatorname{out}(S)$. Then there is a flow $F : E^* \rightsquigarrow \operatorname{out}(S)$, in $G[S]$, where each edge in E^* sends one flow unit, each edge in $\operatorname{out}(S)$ receives at most 0.2 flow units, and the congestion is at most 1.2.*

As before, from the integrality of flow, we can build a set $N = \{\tau_e \mid e \in E^*\}$ of tendrils contained in $G[S]$, where each tendril τ_e connects edge e to some edge of $\operatorname{out}(S)$, each edge in $\operatorname{out}(S)$ participates in at most one tendril, and the total congestion caused by N is at most 2.

Extended Well-Linked Decomposition

We would like to obtain a well-linked decomposition similar to Theorem 18 with one additional property: if $R \in \mathcal{W}$ is a small cluster, then it is $\Omega(1/(\log \log n)^{1.5})$-well-linked.

In order to achieve this, given a set S, we first perform a basic well-linked decomposition as in Theorem 18. Let $\mathcal{W}$ be the resulting decomposition, $E^* = \left(\bigcup_{R \in \mathcal{W}} \operatorname{out}(R)\right) \setminus \operatorname{out}(S)$, and let $F : E^* \rightsquigarrow \operatorname{out}(S)$ be the resulting flow.

For each small cluster $R \in \mathcal{W}$, we then perform another round of basic well-linked decomposition on the set R (notice that now $k' = |\operatorname{out}(R)| \leq L$). Let $\mathcal{W}_R$ be the resulting partition, E^*_R the resulting set of partition edges (excluding the edges of $\operatorname{out}(R)$), and F^R the resulting flow.

We build the final partition $\mathcal{W}'$ as follows. Start from $\mathcal{W}$, and replace each small set $R \in \mathcal{W}$ by the sets in $\mathcal{W}_R$. Let $E^{**} = \left(\bigcup_{R \in \mathcal{W}'} \operatorname{out}(R)\right) \setminus \operatorname{out}(S)$. We extend the flow $F : E^* \rightsquigarrow \operatorname{out}(S)$ to flow $F' : E^{**} \rightsquigarrow \operatorname{out}(S)$ as follows. Consider a small set $R \in \mathcal{W}$. Recall that we have defined a flow $F^R : E^*_R \rightsquigarrow \operatorname{out}(R)$ inside R, where each edge in E^*_R sends one flow unit, and each edge in $\operatorname{out}(R)$ receives at most 0.2 flow units. We concatenate this flow with the flow originating from the edges of $\operatorname{out}(R)$ in F (after scaling it appropriately, by the factor of at most 0.2). After we process all small sets $R \in \mathcal{W}$, we obtain the final flow F'. Every edge in E^{**} now sends one flow unit, every edge in $\operatorname{out}(S)$ receives at most 0.4 flow units, and total congestion is at most 3. As before, from the integrality of flow, we can build a set $N = \{\tau_e \mid e \in E^{**}\}$ of tendrils, where each tendril τ_e connects edge e to some edge of $\operatorname{out}(S)$, each edge in $\operatorname{out}(S)$ participates in at most one tendril, and the total congestion caused by N is at most 3. This concludes the proof of Theorem 6.

D Proofs Omitted from Section 2

D.1 Proof of Lemma 1

We describe an algorithm to find a critical cluster Q. Let G' be a graph obtained from G as follows: subdivide every edge $e \in \operatorname{out}(S)$ with a vertex v_e, and let $T' = \{v_e \mid e \in \operatorname{out}(S)\}$. Graph G' is the sub-graph of G induced by $T' \cup S$.

Throughout the algorithm, we maintain a collection π of disjoint subsets of vertices of S, together with a corresponding contracted graph Z, which is obtained from graph G', by contracting every cluster $C \in \pi$ into a super-node v_C. We say that π is a *good collection of clusters*, iff each cluster $C \in \pi$ is small and has the bandwidth property. The value $W(\pi)$ of the collection π of clusters is the number of edges in the corresponding contracted graph Z. We notice that some vertices of S may not belong to any cluster in π. Our initial collection is $\pi = \emptyset$.

We say that a cluster $S' \subseteq S$ is *canonical* for the collection π iff for every cluster $C \in \pi$, either $C \subseteq S'$, or $C \cap S' = \emptyset$.

Throughout the algorithm, we also maintain an active large cluster $S' \subseteq S$ (the initial cluster $S' = S$), and we will ensure that S' is canonical w.r.t. the current collection π of good clusters, and it has the bandwidth property. We perform a number of iterations. In each iteration, one of the following three things happens: we either find a new good collection π' of clusters, with $W(\pi') < W(\pi)$, or find a critical cluster Q as required, or select a sub-cluster $S'' \subsetneq S$ as our next active cluster. In the latter case, we will guarantee that S'' is canonical for the current collection π of good clusters, and it has the bandwidth property. An execution of an iteration is summarized in the next lemma.

Lemma 7 *Let π be a good collection of clusters, and let $S' \subseteq S$ be a large cluster with the bandwidth property, such that S' is canonical for π. Assume additionally that there is a set $E_{S'} \subseteq \operatorname{out}(S')$ of $L/4$ edges, and a set $\mathcal{P}_{S'} : E_{S'} \rightsquigarrow_{\eta^*} \operatorname{out}(S)$ of paths in graph G, contained in $S \setminus S'$, where each edge in $\operatorname{out}(S)$ is an endpoint of at most one path. Then there is an efficient algorithm, whose output is one of the following:*

- *Either a good collection π' of clusters with $W(\pi') < W(\pi)$.*
- *Or establishes that S' is a critical cluster, by computing, if $|S'| > 1$, a partition π^* of S' into small clusters that have bandwidth property, such that (S', π^*) has the weight property.*
- *Or a sub-cluster $S'' \subsetneq S'$, such that S'' is large, canonical for π, has the bandwidth property, and there is a set $E_{S''} \subseteq \operatorname{out}(S'')$ of $L/4$ edges, and a set $\mathcal{P}_{S''} : E_{S''} \rightsquigarrow_{\eta^*} \operatorname{out}(S)$ of paths in graph G, contained in $S \setminus S''$, where each edge in $\operatorname{out}(S)$ is an endpoint of at most one path.*

Proof: If $|S'| = 1$, then clearly S' is a critical cluster, so we stop the algorithm and return S'. We assume from now on that $|S'| > 1$. Our algorithm consists of two steps. In the first step, we try to find a large canonical

cluster $S'' \subseteq S'$ that has the bandwidth property. In the second step, we either find a subset $E_{S''} \subseteq \text{out}(S'')$ of $L/4$ edges, and a set $\mathcal{P}_{S''} : E_{S''} \rightsquigarrow_{\eta^*} \text{out}(S)$ of paths as required, or compute a new collection π' of good clusters with $W(\pi') < W(\pi)$. We now turn to describe the two steps.

Step 1. We construct a partition π_1 of the vertices of S' into small clusters, as follows. Start with $\pi_1 = \{C \in \pi \mid C \subseteq S'\}$. Let $\tilde{S}'$ be the subset of vertices of S' that do not belong to any cluster of π_1. If any vertex $v \in \tilde{S}'$ has a degree greater than L in graph G, then we set $S'' = \{v\}$. Clearly, S'' is a large cluster with the bandwidth property. We then continue to the second step. Assume now that every vertex $v \in \tilde{S}'$ has degree at most L in G. We then add every vertex of $\tilde{S}'$ as a separate cluster to π_1. That is, the set of the new clusters that we add to π_1 is $\left\{\{v\} \mid v \in \tilde{S}'\right\}$. Observe that π_1 now defines a partition of the vertices of S' into small clusters that all have the bandwidth property.

Our next step is to establish whether (S', π_1) has the weight property. We construct the graph $H_{S'}$, which is obtained from $G[S']$ by contracting every cluster $C \in \pi_1$ into a super-node v_C. We set the weight of v_C be $|\text{out}_G(C)|$. We then run the algorithm $\mathcal{A}_{\text{ARV}}$ to find an approximate sparsest cut (A, B) in graph $H_{S'}$.

If $|E_{H_{S'}}(A, B)| \geq \lambda \cdot \alpha_{\text{ARV}}(n) \cdot \min\left\{\sum_{v \in A} w(v), \sum_{v \in B} w(v)\right\}$, then we are guaranteed that for every partition $(\tilde{A}, \tilde{B})$ of the vertices of $H_{S'}$,

$$|E_{H_{S'}}(\tilde{A}, \tilde{B})| \geq \lambda \cdot \min\left\{\sum_{v \in \tilde{A}} w(v), \sum_{v \in \tilde{B}} w(v)\right\},$$

and so the weight property holds for the current partition π_1. We then return S' as a critical cluster, together with the partition $\pi^* = \pi_1$.

We assume from now on that $|E_{H_{S'}}(A, B)| < \lambda \cdot \alpha_{\text{ARV}}(n) \cdot \min\left\{\sum_{v \in A} w(v), \sum_{v \in B} w(v)\right\}$. Let A' be the subset of vertices obtained from A, after we un-contract all clusters in π_1, and let B' be the set obtained similarly from B, so (A', B') is a partition of S'. Denote $T_A = |\text{out}_G(A') \cap \text{out}_G(S')|$, and $T_B = |\text{out}_G(B') \cap \text{out}_G(S')|$. Recall that there is a one-to-one correspondence between $E_{H_{S'}}(A, B)$ and $E_G(A', B')$. We therefore do not distinguish between these two sets of edges and denote both of them by Γ. We assume w.l.o.g. that $|T_A| \leq |T_B|$. As observed before,

$$|\Gamma| \leq \lambda \cdot \alpha_{\text{ARV}}(n) \cdot \sum_{v \in A} w(v) \tag{4}$$

On the other hand, from the bandwidth property of the cluster S', we get that $|\Gamma| \geq \alpha_{\text{BW}} \cdot |T_A|$, and so $|\text{out}_G(A')| = |\Gamma| + |T_A| \leq \left(1 + \frac{1}{\alpha_{\text{BW}}}\right) \cdot |\Gamma| \leq \frac{2}{\alpha_{\text{BW}}}|\Gamma|$.

Combining this with equation (4), we get that:

$$\sum_{v \in A} w(v) \geq \frac{|\Gamma|}{\lambda \cdot \alpha_{\text{ARV}}(n)} \geq |\text{out}_G(A')| \frac{\alpha_{\text{BW}}}{2\lambda \cdot \alpha_{\text{ARV}}(n)} \geq 4|\text{out}_G(A')| \tag{5}$$

since $\lambda = \frac{\alpha_{\text{BW}}}{8\alpha_{\text{ARV}}(n)}$.

We now construct a new graph H from graph G, as follows. First, we sub-divide every edge $e \in \text{out}_G(A')$ by a vertex v_e, and we let $\mathcal{T}'' = \{v_e \mid e \in \text{out}_G(A')\}$. We then let H be the sub-graph of G induced by $A' \cup \mathcal{T}''$, after we contract all clusters $C \in \pi$ that are contained in A'. Observe that H is identical to $H_{S'}[A]$, except for the edges adjacent to the vertices in $\mathcal{T}''$ that are added in graph H. In particular, $V(H) \setminus \mathcal{T}' = A$. We perform the extended well-linked decomposition of the set A of vertices in graph H, using Theorem 6, obtaining a decomposition $\mathcal{W}'$. We now consider two cases.

Case 1: The first case happens if all clusters in $\mathcal{W}'$ are small. In this case, for each cluster $R \in \mathcal{W}'$, let R' be the set of vertices obtained from R after we un-contract all clusters in π. We run another round of extended well-linked decomposition on each such cluster R', obtaining a partition $\mathcal{W}_{R'}$ of R', and we let $\mathcal{W}^* = \bigcup_{R \in \mathcal{W}'} \mathcal{W}_{R'}$ be the resulting partition of all vertices in A'. Observe that we are guaranteed that every cluster in $\mathcal{W}^*$ is small

and has the bandwidth property. We obtain a new good collection π' of clusters from π, by first removing from π all clusters $C \subseteq A'$, and then adding all clusters from $\mathcal{W}^*$. Since $\sum_{\substack{C \in \pi_1: \\ C \subseteq A'}} |\operatorname{out}_G(C)| \geq 4|\operatorname{out}_G(A')|$, while

$$\sum_{C \in \mathcal{W}^*} |\operatorname{out}(C)| \leq \sum_{R \in \mathcal{W}'} 1.4|\operatorname{out}_H(R)| \leq 1.96|\operatorname{out}_H(A)| = 1.96|\operatorname{out}_G(A')|,$$

we are guaranteed that $W(\pi') < W(\pi)$. We then stop the algorithm and return π'.

Case 2: In the second case, there is at least one large cluster $\tilde{S} \in \mathcal{W}'$. Let S'' be the subset of vertices obtained from $\tilde{S}$ after we un-contract every cluster $C \in \pi_1$. Then $S'' \subsetneq S'$ is a large canonical cluster w.r.t. π. We claim that S'' has the bandwidth property. Assume otherwise, and let (X, Y) be a violating partition of S''. Let $T_X = \operatorname{out}(X) \cap \operatorname{out}(S'')$, $T_Y = \operatorname{out}(Y) \cap \operatorname{out}(S'')$, $E' = E_G(X, Y)$, and assume w.l.o.g. that $|T_X| \leq |T_Y|$. Since (X, Y) is a violating partition, $|E'| < \alpha_{\text{BW}} \cdot |T_X|$. Let $\pi'' \subseteq \pi$ be the collection of clusters contained in S''. We now modify the partition (X, Y) of S'', so that for every cluster $C \in \pi''$, either $C \subseteq X$ or $C \subseteq Y$ holds. Consider any such cluster $C \in \pi''$, and let $E'_C = E_G(C \cap X, C \cap Y)$. We partition the edges in $\operatorname{out}_G(C)$ into four subsets, E_X, E_Y, E_{XY}, and E_{YX}, as follows. Let $(u, v) \in \operatorname{out}_G(C)$ with $u \in C$. If $u, v \in X$, then e is added to E_X; if $u, v \in Y$, then e is added to E_Y; if $u \in X, v \in Y$, then e is added to E_{XY}; otherwise it is added to E_{YX}. If $|E_X| + |E_{XY}| \leq |E_Y| + |E_{YX}|$, then we move all vertices of C to Y, and otherwise we move them to X. Assume w.l.o.g. that $|E_X| + |E_{XY}| \leq |E_Y| + |E_{YX}|$, so we have moved the vertices of C to Y. The only new edges added to the cut are the edges of E_X, and since C is a small cluster with the bandwidth property, $|E'_C| \geq \alpha_S \cdot |E_X|$. We charge the edges of E'_C for the edges of E_X, where the charge to every edge of $E'_C$is at most $1/\alpha_S$.

Let (X', Y') be the final partition, obtained after all clusters $C \in \pi''$ have been processed. Notice that the vertices of $\mathcal{T}''$ do not belong to any cluster $C \in \pi''$, so the partition of $\mathcal{T}''$ induced by (X', Y') is the same as the partition induced by (X, Y). From the above charging scheme, since in the original partition, $|E_G(X, Y)| < \alpha_{\text{BW}} \cdot |T_X|$, in the new partition $|E_G(X', Y')| \leq \frac{1}{\alpha_S}|E_G(X, Y)| < \alpha_W \cdot |T_X|$, since $\alpha_{\text{BW}} = \alpha_S \cdot \alpha_W$. Finally, notice that partition (X', Y') of S'' naturally defines a partition $(\tilde{X}, \tilde{Y})$ of $\tilde{S}$, where for each cluster $C \in \pi''$, $v_C \in \tilde{X}$ iff $C \subseteq X'$. But then $|E_H(\tilde{X}, \tilde{Y})| = |E_G(X', Y')| < \alpha_W \min\left\{|\operatorname{out}_H(\tilde{X}) \cap \operatorname{out}_H(\tilde{S})|, |\operatorname{out}_H(\tilde{Y}) \cap \operatorname{out}_H(\tilde{S})|\right\}$, contradicting the fact that $\tilde{S}$ is α_W-well-linked in H. We conclude that S'' has the bandwidth property.

Step 2. We now assume that we are given a large cluster $S'' \subseteq S'$ that has the bandwidth property, and S'' is canonical for π. The goal of this step is to find a collection $E_{S''} \subseteq \operatorname{out}_G(S'')$ of $L/4$ edges, and the set $\mathcal{P}_{S''}$ of paths as required. If we fail to find such a collection of paths, then we will compute a new good collection π' of clusters, such that $W(\pi') < W(\pi)$.

We set up the following flow network. Let $\tilde{S}$ be the set of vertices of Z, obtained from S'' by replacing every cluster $C \in \pi$ with $C \subseteq S''$, by the super-node v_C. Start with the current contracted graph Z, and contract all vertices of $\tilde{S}$ into the source s. Contract all vertices in T' (recall that these are the vertices $\{v_e \mid e \in \operatorname{out}(S)\}$) into the sink t. Let $\mathcal{N}$ be the resulting flow network. We now try to find a flow F' of value $L/4$ from s to t in $\mathcal{N}$. Assume first that such flow can be found. From the integrality of flow, there is a set $E_{S''} \subseteq \operatorname{out}_G(S'')$ of $L/4$ edges, and a set $\mathcal{P}'$ of $L/4$ edge-disjoint paths, where each path connects a distinct edge of $E_{S''}$ to a distinct vertex of T' in graph Z, and the paths in $\mathcal{P}'$ cause congestion 1 in Z. We now show how to find a flow $F^* : E_{S''} \rightsquigarrow_{\eta^*} \operatorname{out}(S)$ in graph G, where each edge in $\operatorname{out}(S)$ receives at most one flow unit. The flow-paths in F^* will follow the paths in $\mathcal{P}'$, except that we need to specify the routing inside the clusters $C \in \pi$. For each such cluster C, the set $\mathcal{P}'$ of paths defines a set D_C of integral 1-restricted demands on the edges of $\operatorname{out}_G(C)$. Since each such cluster C is α_S-well-linked, and it is a small cluster, this set of demands can be routed inside C with congestion at most $\frac{2\beta_{\text{FCG}}(L)}{\alpha_S} \leq \eta^*$. From the integrality of flow, we can find a set $\mathcal{P}^* : E_{S''} \rightsquigarrow_{\eta^*} \operatorname{out}(S)$ of paths in $G[S \setminus S'']$, where each edge in $\operatorname{out}(S)$ serves as an endpoint of at most one such path. We return $S'', E_{S''}$ and $\mathcal{P}^*$ as our output.

Finally, assume that such flow does not exist. From the min-cut/max-flow theorem, we can find a cut R in graph Z, with $|\operatorname{out}(R)| < L/4$, and $\tilde{S} \subseteq R$. Since S'' is a large cluster, $\sum_{v \in R} |\operatorname{out}_Z(v)| \geq L$.

Let R' be the subset of vertices of G obtained by un-contracting all clusters $C \in \pi$ that are contained in R. We perform the extended well-linked decomposition of R', using Theorem 6. Let $\mathcal{W}''$ be the resulting decomposition. Since $|\operatorname{out}_Z(R)| < L/4$, all clusters in $\mathcal{W}''$ are small. We obtain a new collection π' of good clusters from π, by first removing all clusters contained in R', and then adding the clusters in $\mathcal{W}''$. Since $\sum_{C \in \mathcal{W}''} |\operatorname{out}_G(C)| \leq 2|\operatorname{out}_G(R')| < L/2$, while $\sum_{v \in R} |\operatorname{out}_Z(v)| \geq |\operatorname{out}_G(S'')| > L$, we are guaranteed that $W(\pi') < W(\pi)$. □

We now complete the proof of Lemma 1. We start with $S' = S$ and an initial collection $\pi = \emptyset$. We then iteratively apply Lemma 7 to the current cluster S' and the current partition π. If the lemma returns a good collection π' of clusters, whose value is smaller than the value of π, then we replace π with π', set the current active cluster to be $S' = S$, and continue. Otherwise, if it returns a sub-cluster $S'' \subsetneq S'$, then we replace S' with S'' as the current active cluster and continue. Finally, if it establishes that S' is a critical cluster, then we return S', π^*, the set $E_{S'}$ of edges, and the collection $\mathcal{P}_{S'}$ of paths. It is easy to verify that the algorithm terminates in polynomial time: we partition the algorithm execution into phases. A phase starts with some collection π of clusters, and ends when we obtain a new collection π' with $W(\pi') < W(\pi)$. Clearly, the number of phases is bounded by $|E(G)|$. In each phase, we perform a number of iterations, where in each iteration we start with some active cluster $S' \subseteq S$, and replace it with another cluster $S'' \subsetneq S'$. Clearly, the number of iterations in each phase is bounded by n.

D.2 Proof of Observation 1

Let $\mathcal{G} = \{\mathcal{T}_1, \ldots, \mathcal{T}_r\}$. Our first step is to modify the set $\mathcal{D}$ of demands, so that it does not contain demand pairs that belong to the same set $\mathcal{T}_i$. Specifically, for every pair $(u, v) \in \mathcal{D}$, where $u, v \in \mathcal{T}_i$ for some $1 \leq i \leq r$, we replace the demand (u, v) with a pair of demands (u, x), (v, x), where x is any vertex in set $\mathcal{T}_{i+1}$ (if $i = r$, then x is any vertex in $\mathcal{T}_1$). Let $\mathcal{D}'$ be the resulting set of demands. Clearly, any routing of $\mathcal{D}'$ gives a routing of $\mathcal{D}$ with the same congestion, and if the routing of $\mathcal{D}'$ is integral, so is the corresponding routing of $\mathcal{D}$. It is also easy to see that $\mathcal{D}'$ is $(2\gamma, \mathcal{G})$-restricted.

Our second step is to decompose $\mathcal{D}'$ into 4γ demands $\mathcal{D}_1, \ldots, \mathcal{D}_{4\gamma}$, such that each set $\mathcal{D}_j$ of demands is $(1, \mathcal{G})$-restricted, and $\bigcup_{j=1}^{4\gamma} \mathcal{D}_j = \mathcal{D}'$. We construct a multi-graph graph H with vertices $v_1, \ldots, v_r$ corresponding to the groups $\mathcal{T}_1, \ldots, \mathcal{T}_r$ of $\mathcal{G}$. For every pair $(u, v) \in \mathcal{D}'$, with $u \in \mathcal{T}_i$, $v \in \mathcal{T}_j$, we add an edge (i, j) to graph H. Finding the decomposition $\mathcal{D}_1, \ldots, \mathcal{D}_{4\gamma}$ of the set $\mathcal{D}'$ of demands then amounts to partitioning the edges of H into 4γ matchings. Since the maximum vertex degree in H is at most 2γ, such a decomposition can be found by a simple greedy algorithm.

D.3 Proof of Theorem 9

We build the following auxiliary graph G'. Start from graph G, and subdivide every edge $e \in \operatorname{out}(S)$ by a terminal t_e. Let $\mathcal{T}' = \{t_e \mid e \in \operatorname{out}_G(S)\}$. Graph G' is the sub-graph of G induced by $S \cup \mathcal{T}'$. Instead of routing demands over the edges of $\operatorname{out}(S)$, we can now equivalently route pairs of terminals in $\mathcal{T}'$, in graph G'.

Let T be any spanning tree of G'. We use the standard grouping technique to group the terminals in $\mathcal{T}'$ along the tree T into groups of size at least Z and at most $3Z$. Let $U_1, \ldots, U_r$ denote these groups. Let $\mathcal{G}'$ be the resulting partition of the terminals in $\mathcal{T}'$. This partition defines the final partition $\mathcal{G}$ of the edges in $\operatorname{out}(S)$. Assume now that we are given any $(1, \mathcal{G})$-restricted set $\mathcal{D}$ of demands on the set $\operatorname{out}(S)$ of edges. Let $\mathcal{D}'$ be the corresponding set of demands on the set $\mathcal{T}'$ of terminals. It is enough to prove that the demands in $\mathcal{T}'$ can be routed in graph G' with congestion at most 721. We assume w.l.o.g. that r is even, and that $\mathcal{D}' = \{(t_1, t_2), (t_3, t_4), \ldots, (t_{r-1}, t_r)\}$, where for each $1 \leq i \leq r$, $t_i \in U_i$.

Our first step is to extend the set $\mathcal{D}'$ of demands as follows. For each $1 \leq i \leq r$, let $U_i' \subseteq U_i$ be any subset of exactly Z terminals. For each $1 \leq j \leq r/2$, let M_j be any complete matching between U_{2j-1}' and U_{2j}'. The new set $\mathcal{D}^+$ of demands is $\mathcal{D}^+ = \bigcup_{j=1}^{r/2} M_j$.

Let π be the partition of S into small sub-clusters that have the bandwidth property, such that (S, π) has the weight property. We define another auxiliary graph H', as follows. Start from graph G', and contract every cluster $C \in \pi$ into a super-node v_C. The rest of the proof consists of two steps. First, we show that the set

$\mathcal{D}^+$ of demands can be fractionally routed with small congestion on short paths in graph H'. Next, we use the Lovasz Local Lemma to transform this fractional routing into an integral routing of the set $\mathcal{D}'$ of demands in graph G'.

Step 1: Routing on short paths in H'

Definition 4 *We say that a graph $G=(V,E)$ is an α-expander, iff* $\min_{X:|X|\leq|V|/2}\left\{\frac{|E(X,\overline{X})|}{|X|}\right\}\geq\alpha$

We will use the result of Leighton and Rao [LR99], who show that any demand that is routable on an expander graph with no congestion, can also be routed on relatively short paths with small congestion. In order to use their result, we need to turn H' into a constant-degree expander. We do so as follows.

Recall that the vertices of the graph H' are of two types: terminals of $\mathcal{T}'$, and super-nodes v_C for $C\in\pi$. Moreover, from the weight property, for any partition (A,B) of $V(H')$,

$$|E_{H'}(A,B)|\geq\lambda\cdot\min\left\{\sum_{\substack{v_C\in A:\\ C\in\pi}}d_{H'}(v_C),\sum_{\substack{v_C\in B:\\ C\in\pi}}d_{H'}(v_C)\right\}$$

We process the vertices v_C of H' that correspond to the super-nodes one-by-one. Let v be any such vertex, let d be its degree, and let $e_1,\ldots,e_d$ be the edges adjacent to v. We replace v with a degree-3 expander X_v on d vertices, whose expansion parameter is some constant α'. Each edge $e_1,\ldots,e_d$ now connects to a distinct vertex of X_v. Let H'' denote the graph obtained after each super-node of H' has been processed. Notice that the maximum vertex degree in H'' is bounded by 4, $\mathcal{T}'\subseteq V(H'')$, and any fractional routing of the set $\mathcal{D}^+$ of demands in graph H'' on paths of length at most ℓ gives a routing of the same set of demands in H', with the same congestion, on paths of length at most ℓ. We next show that graph H'' is an α-expander, for $\alpha=\lambda\alpha'/4$.

Claim 2 *Graph H'' is an α-expander, for $\alpha=\lambda\alpha'/4$.*

Proof: Assume otherwise, and let (A,B) be a violating cut, that is, $|E_{H''}(A,B)|<\alpha\cdot\min\{|A|,|B|\}$. Notice that for each terminal $t\in\mathcal{T}'$, there is exactly one vertex $v_t\in V(H'')$ adjacent to t, and we can assume w.l.o.g. that both t and v_t belong to the same set, A or B (otherwise t can be moved to the other set, and the sparsity of the cut will only go down).

We use the cut (A,B) to define a partition (A',B'), where $A',B'\neq\emptyset$, of the vertices of H', and show that $|E_{H'}(A',B')|<\lambda\cdot\min\left\{\sum_{\substack{v_C\in A':\\ C\in\pi}}d_{H'}(v_C),\sum_{\substack{v_C\in B':\\ C\in\pi}}d_{H'}(v_C)\right\}$, thus contradicting the weight property of (S,π).

Partition (A',B') is defined as follows. For each terminal $t\in\mathcal{T}'$, if $t\in A$, then we add t to A'; otherwise it is added to B'. For each super-node v_C, if at least half the vertices of X_{v_C} belong to A, then we add v_C to A'; otherwise we add v_C to B'.

We claim that $|E_{H'}(A',B')|\leq|E_{H''}(A,B)|/\alpha'$. Indeed, consider any super-node $v_C\in V(H')$, and consider the partition (A_{v_C},B_{v_C}) of the vertices of X_{v_C} defined by the partition (A,B), that is, $A_{v_C}=A\cap V(X_{v_C})$, $B_{v_C}=B\cap V(X_{v_C})$. Assume w.l.o.g. that $|A_{v_C}|\leq|B_{v_C}|$. Then the contribution of the edges of X_{v_C} to $E_{H''}(A,B)$ is at least $\alpha'\cdot|A_{v_C}|$. After vertex v_C is processed, we add at most $|A_{v_C}|$ edges to the cut. Therefore,

$$|E_{H'}(A',B')|\leq\frac{|E_{H''}(A,B)|}{\alpha'}\leq\frac{\alpha}{\alpha'}\cdot\min\{|A|,|B|\}=\frac{\lambda}{4}\min\{|A|,|B|\}$$

Assume w.l.o.g. that $\sum_{v_C\in A'}d_{H'}(v_C)\leq\sum_{v_C\in B'}d_{H'}(v_C)$. Consider the set A of vertices of H'', and let $A_1\subseteq A$ be the subset of vertices, that belong to expanders X_{v_C}, where $|V(X_{v_C})\cap A|\leq|V(X_{v_C})\cap B|$. Notice that from the expansion properties of graphs X_{v_C}, $|E_{H''}(A,B)|\geq\alpha'|A_1|$, and so $|A_1|\leq\frac{|E_{H''}(A,B)|}{\alpha'}\leq$

$\frac{\alpha}{\alpha'}|A| \leq \frac{|A|}{8}$. Each non-terminal vertex in $A \setminus A_1$ contributes at least 1 to the summation $\sum_{v_C \in A'} d_{H'}(v_C)$, and for each terminal $t \in \mathcal{T}'$, its unique neighbor belongs to A, so $|\mathcal{T}' \cap A| \leq |A|/2$, and $|A \setminus (A_1 \cup \mathcal{T}')| \geq \frac{3}{8}|A|$. Therefore, $\sum_{v_C \in A'} d_{H'}(v_C) \geq \frac{3}{8}|A|$. We conclude that:

$$|E_{H'}(A', B')| \leq \frac{\lambda}{4}|A| \leq \frac{2\lambda}{3} \sum_{v_C \in A'} d_{H'}(v_C)$$

contradicting the weight property of (S, π). □

The following theorem easily follows from the results of Leighton and Rao [LR99], and its proof can be found in [Chu11].

Theorem 19 *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 collection $\mathcal{P}_{u,v}$ of $m = \Theta(\log n)$ paths of 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.*

We now apply Theorem 19 to the set $\mathcal{D}^+$ of demands in graph H''. For every pair (u, v) of terminals with $D^+(u, v) = 1$, we obtain a set $\mathcal{P}'_{u,v}$ of $m = O(\log n)$ paths in graph H'', of length at most $\ell = O(\log n/\lambda) = O(\log^3 n \operatorname{poly}\log\log n)$ each, and the paths in $\bigcup_{(u,v):D^+(u,v)=1} \mathcal{P}'_{u,v}$ cause congestion at most $\tilde{\eta} = O(\log^2 n/\lambda) = O(\log^4 n \operatorname{poly}\log\log n)$ in graph H''. As observed before, for each pair (u, v) of terminals with $D^+(u, v) = 1$, the set $\mathcal{P}'_{u,v}$ of paths in graph H'' gives a set $\mathcal{P}_{u,v}$ of paths in graph H', connecting u to v, such that the length of each path in $\mathcal{P}_{u,v}$ is at most ℓ. Let $\mathcal{P} = \bigcup_{(u,v):D^+(u,v)=1} \mathcal{P}_{u,v}$. Then the paths in $\mathcal{P}$ cause congestion at most $\tilde{\eta}$ in H'. This concludes the first step of the algorithm.

Step 2: Integral routing in G' For each $1 \leq j \leq r/2$, let $\mathcal{B}_j$ be the union of the sets of paths $\mathcal{P}_{u,v} \subseteq \mathcal{P}$ where $u \in U_{2j-1}, v \in U_{2j}$. We call the set $\mathcal{B}_j$ of paths a *bundle*.

Let $c = 13$. We set $Z = \frac{2\tilde{\eta}^{1+\frac{2}{c-1}} \cdot z^{1+\frac{2}{c-1}} \cdot \ell^{\frac{1}{c-1}}}{m} = O((\log n)^{3+\frac{11}{c-1}} \cdot \operatorname{poly}\log\log n) = O(\log^4 n)$, where z is the parameter from Corollary 1.

For each small cluster $C \in \pi$, let $\mathcal{G}_C$ be the partition of the edges of $\operatorname{out}_G(C)$ guaranteed by Corollary 1. We will select one path from each bundle $\mathcal{B}_j$, such that for each small cluster $C \in \pi$, for each group $U \in \mathcal{G}_C$, at most c of the selected paths contain edges of U. 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 20 ([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 $\mathcal{B}_j$, we randomly choose one of the paths $P_j \in \mathcal{B}_j$. Let x_j be the random variable indicating which path has been chosen from $\mathcal{B}_i$.

For each small cluster $C \in \pi$, for each group $U \in \mathcal{G}_C$, we define a bad event $\beta_{C,U}$ to be the event that at least c of the chosen paths contain edges in U. The set $vbl(\beta_{C,U})$ contains all variables x_j, where at least one path in bundle $\mathcal{B}_j$ contains an edge of U. Since the set $\mathcal{P}$ of paths causes congestion at most $\tilde{\eta}$ in H', and each group contains at most z edges, $|vbl(\beta_{C,U})| \leq \tilde{\eta}z$. The number of potential c-tuples of paths (where we take at most one path from each bundle) containing edges of U is at most $(\tilde{\eta}z)^c$, and each such c-tuple is selected with probability at most $1/(Zm)^c$. Therefore, $\mathbf{Pr}[\beta_{C,U}] \leq \left(\frac{\tilde{\eta}z}{Zm}\right)^c$. We denote this probability by p.

For each bundle $\mathcal{B}_j$, there are mZ paths in the bundle, each path contains ℓ edges, and for each such edge e, there are most two bad events $\beta_{C,U}, \beta_{C',U'}$, where $e \in U$ and $e \in U'$. Therefore, for each bad event $\beta_{C,U}$, $|\Gamma(\beta_{C,U})| \leq |vbl(\beta_{C,U})| \cdot 2mZ\ell \leq 2mZ\ell\tilde{\eta}z$. We denote $D = 2mZ\ell\tilde{\eta}z$.

It now only remains to verify that $e(D+1) \cdot p \leq 1$, or equivalently, $e(2mZ\ell\tilde{\eta}z + 1) \cdot \left(\frac{\tilde{\eta}z}{Zm}\right)^c \leq 1$. This is immediate from the choice of $Z = \frac{2\tilde{\eta}^{1+\frac{2}{c-1}} \cdot z^{1+\frac{2}{c-1}} \cdot \ell^{\frac{1}{c-1}}}{m}$.

Assume now that we have selected a collection $P_1, \ldots, P_{r/2}$ of paths, such that none of the bad events $\beta_{C,U}$ happens. Then for each small cluster $C \in \pi$, the set $P_1, \ldots, P_{r/2}$ of paths define a set D_C of $(c, \mathcal{G}_C)$-restricted demands, which can be routed inside C with congestion at most $60(c-1) = 720$ using Corollary 1. Let $\mathcal{P}' = \left\{P'_1, \ldots, P'_{r/2}\right\}$ denote the resulting set of paths in graph G'. Then for each $1 \leq j \leq r/2$, path P'_j connects some terminal $t'_{2j-1} \in U_{2j-1'}$ to some terminal $t'_{2j} \in U_{2j}$. Moreover, since every edge of H' belongs to some group $U \in \mathcal{G}_C$ of some small cluster $C \in \pi$, the total congestion caused by paths in $\mathcal{P}'$ is bounded by 720. Finally, we extend each path P'_j by connecting t_{2j-1} to t'_{2j-1}, and t'_{2j} to t_{2j} via the spanning tree T. Since each group $U \in \mathcal{G}'$ is associated with a sub-tree T_U of T, and all these sub-trees are edge-disjoint, the resulting set of paths gives an integral routing of the set D' of demands in graph G', with congestion at most 721. This concludes the proof of Theorem 9.

E Proofs Omitted from Section 3

E.1 Proof of Theorem 10

The goal of this section is to provide an efficient algorithm for finding a Q-J-decomposition. For each edge $e \in E^J$, if v is the endpoint of the tendril $\tau(e)$ that belongs to some Q-cluster in $\mathcal{Q}$, then we say that v is the *head* of the tendril $\tau(e)$. We also set $Q^* = \bigcup_{Q \in \mathcal{Q}} Q$.

We build the clusters in $\mathcal{Q}$ gradually. The algorithm performs a number of iterations. We start with $\mathcal{Q} = \mathcal{Q}_0$, and in each iteration, we add a new critical cluster Q to $\mathcal{Q}$. In the last iteration, we produce the set $\mathcal{J}$ of the J-clusters, and their tendrils, as required.

We now proceed to describe an iteration. Let $\mathcal{Q}$ be the set of current Q-clusters, and let $Q^* = \bigcup_{Q \in \mathcal{Q}} Q$. Let $S_0 = V \setminus Q^*$.

We start by performing the extended well-linked decomposition of the set S_0 of vertices, using Theorem 6. Let $\mathcal{W}$ be the resulting decomposition, and N the corresponding set of tendrils. If all sets in $\mathcal{W}$ are small, then we set $\mathcal{J} = \mathcal{W}$, and finish the algorithm, so the current iteration becomes the last one. The output is $(\mathcal{Q}, \mathcal{J})$, and the final set of tendrils is N. We will later show that it has all required properties. Assume now that $\mathcal{W}$ contains at least one large cluster, and denote it by S. Let N_S be the set of tendrils originating at edges of $\mathrm{out}(S)$.

We use Lemma 1 to find a critical sub-cluster $Q \subseteq S$, together with the subset $E_Q \subseteq \mathrm{out}(Q)$ of $L/4$ edges, and the set $\mathcal{P}_S : E_Q \rightsquigarrow_{\eta^*} \mathrm{out}(S)$ of paths, that are contained in $S \setminus Q$. Let $\mathcal{P}'_S : E_Q \rightsquigarrow_{\eta^*} \mathrm{out}(S_0)$ be a collection of paths obtained by concatenating the paths in $\mathcal{P}_S$ with the set N_S of tendrils originating at the edges of $\mathrm{out}(S)$. Notice that each edge of $\mathrm{out}(S_0)$ serves as an endpoint of at most one such path, and $|\mathcal{P}'_S| = L/4$. We then add Q to $\mathcal{Q}$, and continue to the next iteration. This concludes the description of an iteration. Consider the final collections $\mathcal{Q}, \mathcal{J}$ of clusters produced by the algorithm. It is immediate to see that Properties (P1)–(P2) hold for it. We only need to establish Property (P3).

Consider any cut $(S, \overline{S})$ in graph G, such that for each cluster $Q \in \mathcal{Q}$, either $Q \subseteq S$, or $Q \subseteq \overline{S}$, and assume that both S and $\overline{S}$ contain at least one cluster in $\mathcal{Q}$. We say that the vertices of S are red and the vertices of $\overline{S}$ are blue.

If both S and $\overline{S}$ contain clusters from $\mathcal{Q}_0$, then the cut $(S, \overline{S})$ must be large by our initial assumption. Assume w.l.o.g. that all clusters in $\mathcal{Q}_0$ are red. Let Q be the first cluster that has been added to $\mathcal{Q}$ over the course of the algorithm, whose vertices are blue. Recall that we have a set $\mathcal{P}'_Q$ of $L/4$ paths connecting the edges of $\mathrm{out}(Q)$ to the edges of $\mathrm{out}(S_0)$ with congestion at most η^*. Therefore, there must be at least $\frac{L}{4\eta^*}$ edges in the cut, so $(S, \overline{S})$ is a large cut. This concludes the proof of Theorem 10.

E.2 Proof of Theorem 11

Our first step is to construct a joined tendril graph H^*, whose construction is identical to the one given by Andrews [And10], except that we use the new Q-J decomposition. The final graph H is obtained by appropriately sampling the edges of H^*.

E.2.1 Joined Tendril Graph

In this section we define the joined-tendril graph H^*. This definition is identical to the definition used by Andrews [And10], except that we use the new Q-J decomposition. Recall that we are given a set $\mathcal{Q}$ of Q-clusters, and a set $\mathcal{J}$ of J-clusters. We have defined $Q^* = \bigcup_{Q\in\mathcal{Q}} Q$, $E^Q = \bigcup_{Q\in\mathcal{Q}} \operatorname{out}(Q)$, and $E^J = \left(\bigcup_{J\in\mathcal{J}} \operatorname{out}(J)\right) \setminus E^Q$. Recall that we are also given a set $N = \left\{\tau_e \mid e \in E^J\right\}$ of tendrils where path τ_e connects e to some edge in E^Q, each edge in E^Q is an endpoint of at most one tendril, and the total congestion caused by the set N of tendrils is at most 3. Moreover, the tendrils do not use edges whose both endpoints belong to Q^*. We define $E' = \bigcup_{J\in\mathcal{J}} \operatorname{out}(J)$, that is, E' consists of edges in E^J and $\operatorname{out}(Q^*)$. We now extend the set N of tendrils to include tendrils $\tau(e)$ for edges $e \in \operatorname{out}(Q^*)$, where for each such edge, $\tau(e) = (e)$. The resulting set N contains a tendril $\tau(e)$ for each edge $e \in E'$, the total congestion due to the set N of tendrils is at most 3, and each edge in $\operatorname{out}(Q^*)$ serves as an endpoint of at most two such tendrils. We are now ready to define the graph H^*.

The vertices of graph H^* correspond to the Q-clusters, $V(H^*) = \{v_C \mid C \in \mathcal{Q}\}$. The set of the edges of H^* consists of two subsets, E^0 and E^1. For each edge $e = (u, u')$ in G, where $u \in C$, $u' \in C'$ for a pair $C, C' \in \mathcal{Q}$ of distinct Q-clusters, we add the edge $(v_C, v_{C'})$ to E^0. In order to define the set E^1 of edges, we consider the set $\mathcal{J}$ of J-clusters. For each J-cluster $C \in \mathcal{J}$, we define a set E^C of edges, and we eventually set $E^1 = \bigcup_{C\in\mathcal{J}} E^C$. The final set of edges of H^* is $E(H^*) = E^0 \cup E^1$.

Consider some J-cluster $C \in \mathcal{J}$. Let X_C be a degree-3 α'-expander on $|\operatorname{out}(C)|$ vertices, where α' is some constant, and let $f_C : V(X_C) \to \operatorname{out}(C)$ be an arbitrary bijection, mapping each vertex of X_C to an edge of $\operatorname{out}(C)$. For each edge $e = (u, v) \in E(X_C)$, we define an edge $e' \in E^C$, as follows. Consider the edges $e_1 = f_C(u)$, and $e_2 = f_C(v)$, and their tendrils $\tau(e_1), \tau(e_2)$. Let $h_1, h_2 \in Q^*$ be the heads of these two tendrils, respectively, and assume that $h_1 \in C_1, h_2 \in C_2$, where $C_1, C_2 \in \mathcal{Q}$. If $C_1 \neq C_2$, then we add an edge e', connecting v_{C_1} to v_{C_2}, to set E^C. The following notation will be useful later. We denote $p_1(e') = e_1$, and $p_2(e') = e_2$ (the ordering of these two vertices is chosen arbitrarily). We also denote by $\tau_1(e') = \tau(e_1)$, and $\tau_2(e') = \tau(e_2)$, and refer to them as *the first and the second tendrils* of e', respectively. Finally, we denote $f'(e') = e$. Observe that each edge $e'' \in \operatorname{out}(C)$ may serve as $p_1(e')$ or $p_2(e')$ for at most three edges $e' \in E^C$, since the degree of the expander X_C is 3. Notice also that for each edge $e \in E(X_C)$, we add at most one edge e' to E^C. We set $E^1 = \bigcup_{C\in\mathcal{J}} E^C$, and the final set of edges of H^* is $E(H^*) = E^0 \cup E^1$. For consistency, for edges $e \in E^0$, we define the first and the second tendril of e to be the edge e itself. Let $e \in \bigcup_{J\in\mathcal{J}} \operatorname{out}(J)$, and let $\tau(e)$ be its tendril. Assume that $e = (u, v)$, where $u \in C_1$, $v \in C_2$. Notice that $\tau(e)$ may serve as the first or the second tendril of at most six edges in graph H^*: three edges in E^{C_1} (edges e' for which $p_1(e') = e$, or $p_2(e') = e$), and three edges in E^{C_2}. This completes the definition of graph H^*. In the next theorem, we show that the cuts in graph H^* are roughly at least as high as their counterparts in graph G. The theorem and its proof are identical to the ones in [And10].

Theorem 21 *Let (A, B) be any cut in graph H^*. Then there is a cut (A', B') in graph G, such that for each $Q \in \mathcal{Q}$, if $v_Q \in A$, then $Q \subseteq A'$, and if $v_Q \in B$, then $Q \subseteq B'$. Moreover, $|E_G(A', B')| \leq O(|E_{H^*}(A, B)|)$.*

Proof: We will call the vertices of A and B in H^* red and blue, respectively. We will assign the colors, red or blue, to the vertices of G, and then the cut (A', B') will be defined based on the colors of vertices, where A', B' contain vertices whose colors are red and blue respectively.

For each critical cluster $Q \in \mathcal{Q}$, if v_Q is red, then we color all its vertices red, and otherwise we color all its vertices blue. Now, consider some J-cluster $J \in \mathcal{J}$. Recall that each edge $e \in \operatorname{out}(J)$ has a tendril $\tau(e)$ associated with it, that connects e to some vertex in Q^*. This vertex is already colored red or blue. If at least

half the tendrils in set $\{\tau(e) \mid e \in \text{out}(J)\}$ have their head colored red, then we color all vertices of J red; otherwise we color all vertices of J blue. This completes the definition of the cut (A', B') of $V(G)$.

We now show that $|E_G(A', B')| \leq O(|E_{H^*}(A, B)|)$, by defining a mapping from $E_G(A', B')$ to $E_{H^*}(A, B)$, where the number of edges mapped to each edge of $E_{H^*}(A, B)$ is bounded by a constant. We say that an edge of graph H^* or of graph G is multi-colored iff one of its endpoints is red and the other is blue. Consider some multi-colored edge e in graph G. Since the endpoints of this edge have different colors, it connects vertices from two different clusters, C and C'. If both clusters are critical, then we map e to itself. We will ignore such edges from now on, and we let $\hat{E}$ denote the remaining multi-colored edges. If exactly one of the two clusters (say C) is a J-cluster, and the other one is a critical cluster, then we say that cluster C is responsible for the edge e. Assume now that both clusters C, C' are J-clusters, where C is red and C' is blue. If the head of $\tau(e)$ is red, then we say that C' is responsible for e; otherwise we say that C is responsible for e.

So far we have identified, for each edge $e \in \hat{E}$, a cluster $C(e)$ that is responsible for e. This cluster has the property that its color is opposite of the color of the head of $\tau(e)$.

Consider any J-cluster C, and assume that C is responsible for n_C edges. Recall that we have built an expander X_C, and a bijection $f_C : V(X_C) \to \text{out}(C)$. We now show that the number of multi-colored edges in E^C is at least $\Omega(n_C)$.

We color the vertices of the expander X_C as follows. Recall that each vertex $v \in V(X_C)$ corresponds to an edge $e = f(v) \in \text{out}(C)$. The edge e is in turn is connected to Q with a tendril $\tau(e)$, whose head is colored either red or blue. If it is colored red, then we color v red as well, and if it is colored blue, then we color v blue.

Assume w.l.o.g. that C is colored blue. Then for each one of the n_C edges e for which C is responsible, the head of $\tau(e)$ is red, while the majority tendrils in set $\{\tau(e) \mid e \in \text{out}(C)\}$ has a blue endpoints (that is why we have colored C blue). So the red-blue coloring of X_C must define a cut of size at least $\Omega(n_C)$ in graph X_C. If an edge $e \in E(X_C)$ belongs to this cut, then its corresponding edge $e' \in E^C$ must be a multi-colored edge. So the number of multi-colored edges in E^C is at least $\Omega(n_C)$. Overall, the number of multi-colored edges in E^1 is at least $\Omega(|\hat{E}|)$, and the total number of multi-colored edges in H^*, $|E_{H^*}(A, B)| \geq \Omega(|E_G(A', B')|)$ as required. □

Combining Theorem 21 with (Property P3) of the Q-J decomposition, we obtain the following corollary.

Corollary 4 *The value of the minimum cut in graph H^* is $\Omega\left(\frac{L}{\eta^*}\right)$.*

E.2.2 Graph H

We now complete the construction of graph H, with Properties (C1)–(C4). Recall that graph H^* already has the first two properties. The idea is that we will suitably sample some edges of H^* and add them to graph H. This will allow us to (approximately) preserve the cuts, while ensuring Properties (C3) and (C4). In our edge sampling, we use the following theorem, which is a variant of Karger's graph skeleton constructions [Kar99, Kar00, KS96].

Theorem 22 *Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be any graph, where the size of the minimum cut is C. Assume that the edges in $\mathbf{E}$ are partitioned into disjoint subsets $\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_r$, of size at most q each, where $q < C/(128 \ln n)$. Construct a graph $\mathbf{G}' = (\mathbf{V}, \mathbf{E}')$ as follows: for each group $\mathbf{E}_i : 1 \leq i \leq r$, sample one edge $e \in \mathbf{E}_i$ uniformly at random, and add it to $\mathbf{E}'$. Then with probability at least $(1 - 1/n^3)$, for each partition (A, B) of the set V of vertices, $|E_{\mathbf{G}'}(A, B)| \geq \frac{1}{2q}|E_{\mathbf{G}}(A, B)|$.*

Proof: We say that a cut (A, B) is violated if $|E_{\mathbf{G}'}(A, B)| < \frac{1}{2q}|E_{\mathbf{G}}(A, B)|$ Consider some partition (A, B) of the vertices of V, and denote $\hat{E} = E_{\mathbf{G}}(A, B)$. For each $i : 1 \leq i \leq r$, let $\hat{E}_i = \mathbf{E}_i \cap \hat{E}$. For each $1 \leq i \leq r$, we define an indicator random variable y_i, which is set to 1 iff an edge of $\hat{E}_i$ has been added to E'. Then $\mathbf{Pr}[y_i = 1] = |\hat{E}_i|/|\mathbf{E}_i| \geq |\hat{E}_i|/q$, and variables $\{y_i\}_{i=1}^r$ are mutually independent. Then $|E_{\mathbf{G}'}(A, B)| = \sum_i y_i$, and the expectation $\mu = \mathbf{E}[\sum_i y_i] \geq |\hat{E}|/q$. Using the Chernoff bound,

$$\mathbf{Pr}\left[|E_{\mathbf{G}'}(A,B)| < \frac{1}{2q}|E_{\mathbf{G}}(A,B)|\right] = \mathbf{Pr}\left[\sum_i y_i < \mu/2\right] \leq e^{-\mu/8} \leq e^{-|\hat{E}|/(8q)}$$

We use the following theorem of Karger [Kar93]

Theorem 23 (Theorem 6.2 in [Kar93]) *For k half-integer, the number of cuts of weight at most kC in any graph is less than n^{2k}, where C is the size of the minimum cut.*

For each $j : 1 \leq j \leq 2\log n$, we let $\mathcal{C}_j$ denote the collection of all cuts (A,B) with $2^{j-1}C < |E_{\mathbf{G}}(A,B)| \leq 2^jC$. From Karger's theorem, $|\mathcal{C}_j| < n^{2^{j+1}}$. From the above calculations, the probability that a given cut $(A,B) \in \mathcal{C}_j$ is violated, is at most $e^{-2^{j-1}C/(8q)} = e^{-2^{j-4}C/q}$. Using the union bound, the probability that any cut in $\mathcal{C}_j$ is violated is at most

$$\begin{aligned} e^{-2^{j-4}C/q} \cdot n^{2^{j+1}} &= e^{-2^{j-4}C/q + 2^{j+1}\ln n} \\ &\leq e^{-2^{j-5}C/q} \\ &\leq e^{-2^{j+1}\ln n} \end{aligned}$$

if $q < \frac{C}{2^7 \log n}$. Using a union bound over all $1 \leq j \leq 2\log n$, we get that the with probability at least $(1 - 1/n^3)$, none of the cuts is violated. □

We call the procedure outlined in the above theorem an *edge sampling procedure*. Once we specify the partition $\mathbf{E}_1, \ldots, \mathbf{E}_r$ of the edges of our graph, the edge sampling procedure is fixed. We now perform four rounds of the edge sampling procedure, each of which uses a different partition of the edges of H^*. The edges that survive all four rounds of sampling will then be added to graph H.

First partition For each critical cluster $C \in \mathcal{Q}$, let $\mathcal{G}_C$ be the grouping of the edges of $\mathrm{out}(C)$ into groups of size at most $3Z$ guaranteed by Theorem 9. For each cluster $C \in \mathcal{Q}$, for each group $U \in \mathcal{G}_C$, we define a set $S(C,U)$ of edges of H^*, consisting of all edges whose first tendril terminates at an edge of U. Notice that all sets $S(C,U)$ are disjoint. Let $\mathcal{E}_1$ denote the resulting partition of the edges $E(H^*)$. Notice that the size of every group is $O(Z)$, since every edge in $E^{\mathcal{Q}}$ serves as a last edge for at most 3 tendrils, and each tendril may serve as the first tendril of at most 6 edges in H^*.

Second partition The second partition, $\mathcal{E}_2$, is defined exactly like the first partition, except that now we group by the second tendril, and not by the first tendril. Let $\mathcal{E}_2$ denote the resulting grouping.

In order to define the third and the fourth partitions, we consider the J-clusters $C \in \mathcal{J}$. Intuitively, the edges of the expander X_C define demands on the edges of $\mathrm{out}(C)$, that we would like to be able to route inside C with small congestion. We cannot do it directly, and instead employ Corollary 1 for routing in small clusters. For each J-cluster $C \in \mathcal{J}$, let $\mathcal{G}_C$ be the partition of the edges of $\mathrm{out}(C)$ given by Corollary 1. Recall that the size of each group is $z = \mathrm{poly}\log\log n$, and any set of $(\gamma, \mathcal{G}_C)$-restricted integral demands on $\mathrm{out}(C)$ can be routed integrally with constant congestion inside C.

Third partition The third partition, $\mathcal{E}_3$ is defined as follows. Every edge $e \in E^0$ is placed in a separate group. For each J-cluster $C \in \mathcal{J}$, we group the edges of E^C as follows. For each set $U \in \mathcal{G}_C$, we create a group $S(C,U) \subseteq E^C$, that contains all edges $e' \in E^C$, for which $p_1(e') \in U$. This defines a partition of edges of E^C into groups of size $\mathrm{poly}\log\log n$. Each such group is then added to $\mathcal{E}_3$.

Fourth partition The fourth partition, $\mathcal{E}_4$, is defined exactly like the third partition, except that we are using $p_2(e')$ instead of $p_1(e')$ to group the edges of E^C.

We perform the edge sampling procedure four times, once for each partition $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4$. Edges that are selected by all four sampling procedures then become the edges of H. The sizes of groups in partitions $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$ and $\mathcal{E}_4$ are $O(\log^4 n)$, $O(\log^4 n)$, $\mathrm{poly}\log\log n$ and $\mathrm{poly}\log\log n$, respectively.

From Theorem 22, w.h.p., for each partition (A,B) of $V(H)=V(H^*)$, $|E_H(A,B)|=\Omega\left(\frac{|E_{H^*}(A,B)|}{\log^8 n\operatorname{poly}\log\log n}\right)$. Combining this with Theorem 21 and Corollary 4 immediately implies Properties (C1) and (C2). Our sampling using the partitions $\mathcal{E}_1$ and $\mathcal{E}_2$ immediately implies Property (C4). In order to establish Property (C3), consider any edge $e=(v_C,v_{C'})$ in graph H. If e connects two critical clusters, then we define $P_e=\{e\}$. Otherwise, assume that $e\in E^{C''}$ for some J-cluster C''. Let $\hat{E}$ be the subset of edges of $E^{C''}$ that belong to H. Then these edges define a set of $(2,\mathcal{G}_{C''})$-restricted integral demands on the edges of $\operatorname{out}_G(C'')$. From Corollary 1, we can route these demands inside C'' with constant congestion. For each edge $e\in\hat{E}$, let $P_{C''}(e)$ be the corresponding path in this routing.

Consider now some edge $e\in\hat{E}$. We define P_e to be the concatenation of $\tau_1(e)$, $P_{C''}(e)$, and $\tau_2(e)$. Since all tendrils cause a constant congestion in G, and for each J-cluster C'', the routing of the demands on $\operatorname{out}_G(C'')$ is performed with constant congestion, the final congestion of the set $\mathcal{P}_H^E=\{P_e\mid e\in E(H)\}$ of paths is bounded by a constant.

F Proofs Omitted from Section 4

F.1 Proof of Theorem 13

We assume that we are given a collection $\mathcal{M}\subseteq\mathcal{T}\times\mathcal{T}$ of terminal pairs and a value $D>0$, such that for each pair $(t,t')\in\mathcal{M}$, $D(t,t')=1$, and all other demands are 0.

We start by applying Theorem 11 to graph G, where we use the threshold L from the statement of Theorem 13 for the definition of small clusters, and the initial set of critical clusters is $\mathcal{Q}_0=\{\{t\}\mid t\in\mathcal{T}\}$. It is easy to see that each cluster in $\mathcal{Q}_0$ is a critical cluster, and any cut separating the clusters in $\mathcal{Q}_0$ in graph G is large. Let H be the resulting graph guaranteed by Theorem 11.

Since every terminal in $\mathcal{T}$ is mapped to a separate vertex of H, we can view $\mathcal{D}$ as a set of demands for graph H. We now focus on finding a solution to the ICF problem instance in graph H, and later transform it into a solution in the original graph G. We use the following theorem, due to Rao and Zhou [RZ10].

Theorem 24 ([RZ10]) *Let G' be any n-vertex graph, and let $\mathcal{M}'=\{(s_1,t_1),\ldots,(s_k,t_k)\}$ be any set of source-sink pairs in G'. Assume further that the value of the global minimum cut in graph G' is at least $L_{\text{RZ}}=\Omega(\log^5 n)$, and there is a fractional multi-commodity flow, where for each $1\leq i\leq k$, source s_i sends $f_i\leq 1$ flow units to sink t_i, $\sum_{i=1}^k f_i=F$, and the flow causes no congestion in G'. Then there is an efficient randomized algorithm that w.h.p. finds a collection $\mathcal{P}^*$ of at least F/α_{RZ} edge-disjoint paths connecting the terminal pairs in $\mathcal{M}'$, where $\alpha_{\text{RZ}}=O(\log^{10} n)$, and each source-sink pair in $\mathcal{M}'$ is connected by at most one path in $\mathcal{P}^*$.*

Let $x=8\alpha_{\text{RZ}}\cdot\log n=O(\log^{11} n)$. We set $L=2\alpha^*\cdot x\cdot L_{\text{RZ}}=O(\log^{24} n\operatorname{poly}\log\log n)$.

We split graph H into x graphs $H_1,\ldots,H_x$, as follows. For each $1\leq i\leq x$, we set $V(H_i)=V(H)$. In order to define the edge sets of graphs H_i, each edge $e\in E$, chooses an index $1\leq i\leq x$ independently uniformly at random, and is then added to $E(H_i)$. This completes the definition of the graphs H_i. Given any partition (A,B) of the vertices of $V(H)$, let $\operatorname{cut}_G(A,B)$ denote the value of the minimum cut $|E_G(A',B')|$ in graph G, such that for each $v_Q\in A$, $Q\subseteq A'$, and for each $v_Q\in B$, $Q\subseteq B'$. Recall that Theorem 11 guarantees that the size of the minimum cut in H is at least L/α^*, and for each partition (A,B) of $V(H)$, $\operatorname{cut}_G(A,B)\leq\alpha^*\cdot|E_H(A,B)|$. From Theorem 12, w.h.p., for each $1\leq i\leq x$, we have that:

- The value of the minimum cut in H_i is at least $\frac{L}{2\alpha^* x}=L_{\text{RZ}}$.
- For any partition (A,B) of $V(H_i)$, $|E_{H_i}(A,B)|\geq\frac{\operatorname{cut}_G(A,B)}{2x\alpha^*}$.

From now on we assume that both properties hold for each graph H_i. We then obtain the following observation.

Observation 2 *For each $1\leq i\leq x$, there is a fractional solution to the instance $(H_i,\mathcal{D})$ of* basic-ICF *of value $\frac{\lambda_{OPT}}{2x\alpha^*\beta_{\text{FCG}}}$ and no congestion.*

Proof: Assume otherwise. Then the value of the maximum concurrent flow in graph H_i for the set $\mathcal{D}$ of demands is less than $\frac{\lambda_{\text{OPT}}}{2x\alpha^*\beta_{\text{FCG}}}$. We set up an instance of the non-uniform sparsest cut problem on graph H_i with the set $\mathcal{D}$ of demands. Then there is a cut (A,B) in graph H_i, with $\frac{|E_{H_i}(A,B)|}{D_{H_i}(A,B)} < \frac{\lambda_{\text{OPT}}}{2x\alpha^*}$. Let (A',B') be the minimum cut in graph G, where for each $v_Q \in A$, $Q \subseteq A'$, and for each $v_Q \in B$, $Q \subseteq B'$. Then $|E_G(A',B')| = \text{cut}_G(A,B) \leq 2x\alpha^*|E_{H_i}(A,B)|$, while $D_G(A',B') = D_{H_i}(A,B)$. Therefore, $\frac{|E_G(A',B')|}{D_G(A',B')} \leq 2x\alpha^* \frac{|E_{H_i}(A,B)|}{D_{H_i}(A,B)} < \lambda_{\text{OPT}}$. This is impossible, since we have assumed that the value of the optimal fractional solution to the ICF instance $(G,\mathcal{D})$ is λ_{OPT}. □

In the rest of the algorithm, we apply the algorithm of Rao-Zhou to each of the graphs $H_1,\ldots,H_x$ in turn. For each $1 \leq i \leq x$, in the ith iteration we define a subset $\mathcal{M}_i \subseteq \mathcal{M}$ of pairs of terminals (that are not satisfied yet), and we define the set $\mathcal{D}_i$ of demands to be $D_i(t,t') = D$ if $(t,t') \in \mathcal{M}_i$, and $D_i(t,t') = 0$ otherwise. For the first iteration, $\mathcal{M}_1 = \mathcal{M}$. We now describe an execution of iteration $i \geq 1$.

Suppose we are given a set $\mathcal{M}_i$ of terminal pairs and a corresponding set $\mathcal{D}_i$ of demands. We construct a new collection $\mathcal{M}_i'$ of source-sink pairs, where the demand for each pair is 1, as follows. For each pair $(t,t') \in \mathcal{M}_i$, we add $N = \lfloor \frac{\lambda_{\text{OPT}}}{2x\alpha^*\beta_{\text{FCG}}} \cdot D \rfloor$ copies of the pair (t,t') to $\mathcal{M}_i'$. We then apply Theorem 24 to the resulting graph and the set $\mathcal{M}_i'$ of demand pairs. From Observation 2, there is a flow of value at least $F_i = N \cdot |\mathcal{M}_i| = |\mathcal{M}_i'|$ in the resulting graph. Therefore, from Theorem 24, w.h.p. we obtain a collection $\mathcal{P}_i$ of paths connecting the demand pairs in $\mathcal{M}_i$ with no congestion, and $|\mathcal{P}_i| \geq \frac{F_i}{\alpha_{\text{RZ}}} \geq \frac{N \cdot |\mathcal{M}_i|}{\alpha_{\text{RZ}}}$. We say that a pair $(t,t') \in \mathcal{M}_i$ of terminals is satisfied in iteration i, iff the number of paths in $\mathcal{P}_i$ connecting t to t' is at least $\frac{N}{2\alpha_{\text{RZ}}}$. We then let $\mathcal{M}_{i+1} \subseteq \mathcal{M}_i$ be the subset of terminal pairs that are not satisfied in iteration i. This concludes the description of our algorithm for routing on graph H. The key in its analysis is the following simple claim.

Claim 3 *For each* $1 \leq i \leq x$, $|\mathcal{M}_{i+1}| \leq \left(1 - \frac{1}{2\alpha_{\text{RZ}}}\right)|\mathcal{M}_i|$

Proof: Let $\mathcal{M}_i^* \subseteq \mathcal{M}_i$ be the subset of demand pairs that are satisfied in iteration i. It is enough to prove that $|\mathcal{M}_i^*| \geq \frac{1}{2\alpha_{\text{RZ}}}|\mathcal{M}_i|$. Assume otherwise. A pair $(t,t') \in \mathcal{M}_i^*$ contributes at most N paths to $\mathcal{P}_i$, while a pair $(t,t') \in \mathcal{M}_i \setminus \mathcal{M}_i^*$ contributes less than $\frac{N}{2\alpha_{\text{RZ}}}$ paths. Therefore, if $|\mathcal{M}_i^*| < \frac{1}{2\alpha_{\text{RZ}}}|\mathcal{M}_i|$, then:

$$|\mathcal{P}_i| < |\mathcal{M}_i^*| \cdot N + |\mathcal{M}_i \setminus \mathcal{M}_i^*| \cdot \frac{N}{2\alpha_{\text{RZ}}} < \frac{N}{\alpha_{\text{RZ}}}|\mathcal{M}_i|,$$

a contradiction. □

Therefore, after $x = 8\alpha_{\text{RZ}} \cdot \log n$ iterations, we will obtain $\mathcal{M}_{x+1} = \emptyset$, and all demand pairs are satisfied. Recall that a demand pair is satisfied iff there are at least $\frac{N}{2\alpha_{\text{RZ}}} = \Omega\left(\frac{\lambda_{\text{OPT}}}{\alpha_{\text{RZ}}x\alpha^*\beta_{\text{FCG}}} \cdot D\right) = \Omega\left(\frac{\lambda_{\text{OPT}}}{\log^{30} n \operatorname{poly}\log\log n} D\right)$ paths connecting them. Therefore, we have shown an integral solution to the ICF instance $(H,\mathcal{D})$ of value $\Omega\left(\frac{\lambda_{\text{OPT}}}{\log^{30} n \operatorname{poly}\log\log n}\right)$ and no congestion.

We now show how to obtain an integral solution to the ICF instance $(G,\mathcal{D})$, of the same value and constant congestion. Let $\mathcal{P}^*$ be the set of paths in graph H that we have obtained. We transform each path $P \in \mathcal{P}^*$ into a path P' connecting the same pair of terminals in graph G. Recall that all terminals in $\mathcal{T}$ are vertices in both G and H. For each edge $e = (v_Q, v_{Q'})$ on path P, we replace e with the path P_e, connecting some vertex $u \in Q$ to some vertex $u' \in Q'$, guaranteed by Property (C3) of graph H. Once we process all edges on all paths $P \in \mathcal{P}^*$, we obtain, for each cluster $Q \in \mathcal{Q}$, a set $\mathcal{D}_Q$ of demands on the edges of $\text{out}(Q)$, that need to be routed inside the cluster Q. From Property (C4), this set of demands must be $(2,\mathcal{G}_Q)$-restricted. Combining Observation 1 with Theorem 9, we obtain an efficient randomized algorithm that w.h.p. routes the set $\mathcal{D}_Q$ of demands integrally inside Q with constant congestion. For each path $P \in \mathcal{P}^*$, we can now combine the paths P_e for $e \in P$ with the resulting routing inside the clusters Q for each $v_Q \in P$ to obtain a path P' in graph G connecting the same pair of terminals as P. Since the set $\{P_e \mid e \in E(H)\}$ of paths causes a constant congestion in graph G from Property (C3), the resulting set of paths causes a constant congestion in G.

F.2 Proof of Lemma 2

We start with $S = V(G)$, and then perform a number of iterations. Let $G' = G[S]$, and let $\mathcal{T}_S = \mathcal{T} \cap S$. Let (A, B) be the minimum cut separating the terminals in $\mathcal{T}_S$ in graph G', and assume w.l.o.g. that $|A \cap \mathcal{T}_S| \leq |B \cap \mathcal{T}_S|$. If $|E_{G'}(A, B)| < L$, then we set $S = A$, and continue to the next iteration. Otherwise, we output S as a good set. (If $S = V(G)$, then we declare that every cut separating the terminals in $\mathcal{T}$ has value at least L.)

Clearly, if S is the final set that the algorithm outputs, then every cut in graph $G[S]$ separating the set $\mathcal{T} \cap S$ of terminals has value at least L. It only remains to show that $|\operatorname{out}_G(S)| \leq L \log k$. Since $|\mathcal{T}| \leq k$, and the number of terminals contained in set S goes down by a factor of at least 2 in every iteration, there are at most $\log k$ iterations. In each iteration, at most L edges are deleted. Therefore, $|\operatorname{out}_G(S)| \leq L \log k$.

F.3 Proof of Theorem 14

We use the following lemma as a subroutine.

Lemma 8 *Let G' be any graph, and let $\mathcal{S}_1, \mathcal{S}_2$ be two sets of paths in G', where the paths in each set are edge-disjoint (but the paths in $\mathcal{S}_1 \cup \mathcal{S}_2$ may share edges). Assume further that all paths in $\mathcal{S}_1$ originate at the same vertex s. Then we can efficiently find a subset $\mathcal{S}'_1 \subseteq \mathcal{S}_1$ of at least $|\mathcal{S}_1| - 2|\mathcal{S}_2|$ paths, and for each path $P \in \mathcal{S}_2$, another path $\tilde{P}$ connecting the same pair of vertices as P, such that, if we denote $\mathcal{S}'_2 = \left\{\tilde{P} \mid P \in \mathcal{S}_2\right\}$, then:*

1. *All paths in $\mathcal{S}'_1 \cup \mathcal{S}'_2$ are edge-disjoint.*
2. *Let E' and $\tilde{E}$ be the sets of edges used by at least one path in $\mathcal{S}_1 \cup \mathcal{S}_2$ and $\mathcal{S}'_1 \cup \mathcal{S}'_2$ respectively. Then $\tilde{E} \subseteq E'$.*

In other words, the lemma re-routes the paths in $\mathcal{S}_2$, using the paths in $\mathcal{S}_1$, and then sets $\mathcal{S}'_1$ to be the subset of paths in $\mathcal{S}_1$ that do not share edges with the new re-routed paths in $\mathcal{S}'_2$.

The proof of Lemma 8 appears below. We now complete the proof of Theorem 14 using this lemma. We build a graph G' from graph G, by replacing each edge of G with γ parallel edges. It is enough to define the subsets $\mathcal{P}^*_i \subseteq \mathcal{P}_i$, and the paths $\tilde{P}$ for $P \in \mathcal{P}''$ in graph G', such that, in the resulting set $\tilde{\mathcal{P}}' \cup \tilde{\mathcal{P}}''$, all paths are edge-disjoint. From now on we focus on finding these path sets in graph G'.

We perform k' iterations, where in iteration i we process the paths in set $\mathcal{P}_i$, and define a subset $\mathcal{P}^*_i \subseteq \mathcal{P}_i$. In each iteration, we may also change the paths in $\mathcal{P}''$, by replacing some of its paths with paths that have the same endpoints as the original paths (we call this process re-routing). We maintain the invariant that at the beginning of iteration i, the paths in set $\mathcal{P}'' \cup \left(\bigcup_{i'=1}^{i-1} \mathcal{P}^*_{i'}\right)$ are edge-disjoint in G'. We now describe the ith iteration, for $1 \leq i \leq k'$.

Let $\mathcal{S}_1 = \mathcal{P}_i$, and let $\mathcal{S}_2$ be the collection of consecutive segments of paths in $\mathcal{P}_2$ that are contained in S. The sets $\mathcal{S}_1$ and $\mathcal{S}_2$ of paths are both edge-disjoint in graph G', and so we can apply Lemma 8 to them, obtaining the sets $\mathcal{S}'_1$ and $\mathcal{S}'_2$ of paths. We then set $\mathcal{P}^*_i = \mathcal{S}'_1$, and modify every path $P'' \in \mathcal{P}_2$ by removing the segments of $\mathcal{S}_2$ from it, and adding the corresponding segments of $\mathcal{S}'_2$ instead. Let $\mathcal{P}''$ be the collection of the resulting paths. Clearly, the paths in $\mathcal{P}''$ connect the same pairs of terminals as the original paths, and they continue to be edge-disjoint in G' (since the re-routing was only performed inside the graph $G'[S]$). Moreover, since the paths in all sets $\mathcal{P}_1, \ldots, \mathcal{P}_i$ are edge-disjoint, and we have only used the edges of the paths in $\mathcal{P}_i$ to re-route the paths in $\mathcal{P}''$, the paths in set $\mathcal{P}'' \cup \left(\bigcup_{i'=1}^{i} \mathcal{P}^*_{i'}\right)$ are edge-disjoint in G'. Finally, observe that $|\mathcal{S}_2| \leq \gamma L \log n$, since $|\operatorname{out}_G(S)| \leq L \log n$, and every path in $\mathcal{S}_2$ contains at least one edge of $\operatorname{out}_{G'}(S)$. Therefore, $|\mathcal{P}^*_i| \geq |\mathcal{P}_i| - 2L\gamma \log n$. Once we process all sets $\mathcal{P}_1, \ldots \mathcal{P}_{k'}$, our output is $\{\mathcal{P}^*_i\}_{i=1}^{k'}$, and we output the final set $\mathcal{P}''$ that contains a path $\tilde{P}$ for each path P in the original set.

It now only remains to prove Lemma 8.

Proof: [Of Lemma 8]

The proof is very similar to arguments used by Conforti et al. [CHR03]. Given any pair (P, P') of paths, we say that paths P and P' intersect at edge e, if both paths contain edge e, and we say that P and P' intersect iff they share any edge.

We set up an instance of the stable matching problem in a multi-graph. In this problem, we are given a complete bipartite multigraph $G = (A, B, E)$, where $|A| = |B|$. Each vertex $v \in A \cup B$ specifies an ordering R_v of the edges adjacent to v in G. A complete matching M between the vertices of A and B is called stable iff, for every edge $e = (a, b) \in E \setminus M$, the following holds. Let e_a, e_b be the edges adjacent to a and b respectively in M. Then either a prefers e_a over e, or b prefers e_b over b. Conforti et al. [CHR03], generalizing the famous theorem of Gale and Shapley [GS62], show an efficient algorithm to find a complete stable matching M in any such multigraph.

Given the sets $\mathcal{S}_1, \mathcal{S}_2$ of paths, we set up an instance of the stable matching problem as follows. Set A contains a vertex $a(P)$ for each path $P \in \mathcal{S}_1$. For each path $P \in \mathcal{S}_2$, if x, y are the two endpoints of P, then we add two vertices $b(P, x)$ and $b(P, y)$ to B. In order to ensure that $|A| = |B|$, we add dummy vertices to B as needed.

For each pair $P \in \mathcal{S}_1$, $P' \in \mathcal{S}_2$ of paths, for each edge e that these paths share, we add two edges $(a(P), b(P', x))$ and $(a(P), b(P', y))$, where x and y are the endpoints of P', and we think of these new edges as representing the edge e. We add additional dummy edges as needed to turn the graph into a complete bipartite graph.

Finally, we define preference lists for vertices in A and B. For each vertex $a(P) \in A$, the edges incident to $a(P)$ are ordered according to the order in which they appear on path P, starting from s. The dummy edges incident to $a(P)$ are ordered arbitrarily at the end of the list.

Consider now some vertex $b(P, x) \in B$. We again order the edges incident to $b(P, x)$ according to the order in which their corresponding edges appear on the path P, when we traverse P starting from x. The dummy edges incident on $b(P, x)$ are added at the end of the list in an arbitrary order. The preference list of the vertex $b(P, y)$ is defined similarly, except that now we traverse P starting from y. Finally, the preference lists of the dummy vertices are arbitrary.

Let M be any perfect stable matching in the resulting graph. We let $\mathcal{S}'_1 \subseteq \mathcal{S}_1$ be the subset of paths that are matched to the dummy vertices. Clearly, $|\mathcal{S}'_1| \geq |\mathcal{S}_1| - 2|\mathcal{S}_2|$. For each path $P \in \mathcal{S}_2$, we now define a path $\tilde{P}$, as follows. Let x, y be the two endpoints of P. If at least one of the vertices $b(P, x), b(P, y)$ participates in M via a dummy edge, then we let $\tilde{P} = P$, and we say that P is of type 1. Otherwise, let e, e' be the edges of M incident on $b(P, x)$ and $b(P, y)$, respectively, and let $P_1, P_2 \in \mathcal{S}_1$ be two paths such that $a(P_1)$ is the other endpoint of e and $a(P_2)$ is the other endpoint of e'. Let e_1, e_2 be the edges of the original graph that the edges e, e' represent. Let $\sigma_1(P)$ be the segment of P from x to e; $\sigma_2(P)$ the segment of P_1 from e to s; $\sigma_3(P)$ the segment of P_2 from s to e'; and $\sigma_4(P)$ the segment of P from e' to y. We set $\tilde{P}$ be the concatenation of $\sigma_1(P), \sigma_2(P), \sigma_3(P), \sigma_4(P)$, and we say that P is of type 2. Let $\mathcal{S}'_2 = \left\{ \tilde{P} \mid P \in \mathcal{S}_2 \right\}$. It now only remains to show that all paths in $\mathcal{S}'_1 \cup \mathcal{S}'_2$ are edge-disjoint. It is immediate that the paths in $\mathcal{S}'_1$ are edge-disjoint, since the paths in $\mathcal{S}_1$ were edge-disjoint. We complete the proof in the following two claims.

Claim 4 *All paths in $\mathcal{S}'_2$ are edge-disjoint.*

Proof: Assume otherwise, and let $\tilde{P}, \tilde{P}' \in \mathcal{S}'_2$ be any pair of paths that share an edge, say e. First, it is impossible that both P and P' are of type 1, since then $P, P' \in \mathcal{S}_2$, and so they must be edge-disjoint. So at least one of the two paths must be of type 2. Assume w.l.o.g. that it is P, and consider the four segments $\sigma_1(P), \sigma_2(P), \sigma_3(P), \sigma_4(P)$ of $\tilde{P}$, and the two edges e_1, e_2 that we have defined above. Let P_1 and P_2 be the paths in $\mathcal{S}_1$ on which the segments $\sigma_2(P), \sigma_3(P)$ lie.

If P' is of type 1, then it can only intersect $\sigma_2(P)$ or $\sigma_3(P)$, as the paths P and P' are edge-disjoint. Assume w.l.o.g. that P' intersects $\sigma_2(P)$, and let e' be any edge that they share. Let x' be the endpoint of P' such that the edge incident to $b(P', x')$ in M is a dummy edge. Then $b(P', x')$ prefers e' to its current matching, and $a(P_1)$ prefers e' to its current matching as well, as e' lies before e_1 on path P', a contradiction.

Assume now that P' is of type 2, and consider the segments $\sigma_1(P'), \sigma_2(P'), \sigma_3(P'), \sigma_4(P')$ of $\tilde{P}'$. Since the sets $\mathcal{S}_1, \mathcal{S}_2$ of paths are edge-disjoint, the only possibilities are that either one of the segments $\sigma_1(P), \sigma_4(P)$ intersects one of the segments $\sigma_2(P'), \sigma_3(P')$, or one of the segments $\sigma_1(P'), \sigma_4(P')$ intersects one of the segments $\sigma_2(P), \sigma_3(P)$. Assume w.l.o.g. that $\sigma_1(P)$ shares an edge e with $\sigma_2(P')$. Let x be the endpoint of P to which $\sigma_1(P)$ is adjacent, and let e_1 be the last edge on $\sigma_1(P)$, and let $P_1 \in \mathcal{S}_1$ be the path that shares e_1 with P. Then vertex $b(P, x)$ prefers the edge e to its current matching, as it appears before e_1 on P, starting from x. Similarly, $a(P_1)$ prefers e to its current matching, a contradiction. □

Claim 5 *Paths in $\mathcal{S}_1'$ and $\mathcal{S}_2'$ are edge-disjoint from each other.*

Proof: Assume otherwise, and let $\tilde{P} \in \mathcal{S}_1'$, $\tilde{P}' \in \mathcal{S}_2'$ be two paths that share an edge e. It is impossible that $\tilde{P}'$ is of type 1: otherwise, for some endpoint x of P', $b(P', x)$ is adjacent to a dummy edge in M, and $a(P)$ is also adjacent to a dummy edge in M, while both of them prefer e, a contradiction. Therefore, P' is of type 2. Consider the four segments $\sigma_1(P'), \sigma_2(P'), \sigma_3(P'), \sigma_4(P')$. Since the paths in $\mathcal{S}_1$ and $\mathcal{S}_2$ are edge-disjoint, the only possibility is that e belongs to either $\sigma_1(P')$, or to $\sigma_4(P')$. Assume w.l.o.g. that $e \in \sigma_1(P')$. Let x be the endpoint of P' to which $\sigma_1(P')$ is adjacent. Then $b(P', x)$ prefers e to its current matching, and similarly $a(P)$ prefers e to its current matching, a contradiction. □ □

F.4 Proof of Lemma 3

The proof is by induction on the recursion depth. In the base case, when no cut of size less than L separates the terminals of $\mathcal{T}$ in G, the correctness follows directly from Theorem 13.

Otherwise, consider the set $\mathcal{P}'$ of paths. For each pair $(s,t) \in \mathcal{M}_S$, let $\mathcal{P}'(s,t) \subseteq \mathcal{P}'$ be the subset of paths connecting s to t in $\mathcal{P}'$. From Theorem 13, we are guaranteed that $|\mathcal{P}'(s,t)| \geq \lfloor \frac{\lambda_{\mathsf{OPT}}}{\beta} D \rfloor$ of each $(s,t) \in \mathcal{M}_S$, and the paths in $\mathcal{P}'$ cause congestion at most γ in $G[S]$.

Consider now the set $\mathcal{P}''$ of paths. For each pair $(s,t) \in \mathcal{M}'$, let $\mathcal{P}''(s,t) \subseteq \mathcal{P}''$ be the subset of paths connecting s to t in $\mathcal{P}''$. From the induction hypothesis, $|\mathcal{P}''(s,t)| \geq \lfloor \frac{\lambda_{\mathsf{OPT}}}{4\beta} D \rfloor$ for all $(s,t) \in \mathcal{M}'$, and the paths in $\mathcal{P}''$ cause congestion at most γ in G.

Consider now the final set $\mathcal{P} = \tilde{\mathcal{P}}' \cup \tilde{\mathcal{P}}''$ of paths returned by the algorithm. From Theorem 14, the paths in $\mathcal{P}$ cause congestion at most γ, as required. For each pair $(s,t) \in \mathcal{M}_S$, the set $\tilde{\mathcal{P}}'$ of paths contains at least $\lfloor \frac{\lambda_{\mathsf{OPT}}}{\beta} D \rfloor - 2L\gamma \log n \geq \lfloor \frac{\lambda_{\mathsf{OPT}}}{4\beta} D \rfloor$ paths. For each pair $(s,t) \in \mathcal{M}_S$, if $\tilde{\mathcal{P}}''(s,t)$ is the subset of paths of $\tilde{\mathcal{P}}''$ connecting s to t, then $|\tilde{\mathcal{P}}''(s,t)| = |\mathcal{P}''(s,t)| \geq \lfloor \frac{\lambda_{\mathsf{OPT}}}{4\beta} D \rfloor$, since each path in $\mathcal{P}''$ is replaced by a path connecting the same pair of vertices in $\tilde{\mathcal{P}}''$. Therefore, each pair $(s,t) \in \mathcal{M}$ is connected by at least $\lfloor \frac{\lambda_{\mathsf{OPT}}}{4\beta} D \rfloor$ paths in $\mathcal{P}$.

G Proof Omitted from Section 6

G.1 Proof of Theorem 16

Given an instance of the MMIS problem, we construct an instance of group-ICF on a line graph as follows. Let $\mathcal{J} = \{1, \ldots, n\}$ be the input set of jobs, and let $\mathcal{I} = \bigcup_{j=1}^{n} \mathcal{I}_j$. Let $\mathcal{S}$ be the set of endpoints of all intervals in $\mathcal{I}$. We create a line graph $G = (V, E)$, whose vertex set is $\mathcal{S}$, and the vertices are connected in the order in which they appear on the time line. Clearly, $|V(G)| \leq 2N(\mathcal{J})$. For each job j, we create a demand pair (S_j, T_j), as follows. Assume w.l.o.g. that $\mathcal{I}_j = \{I_1, I_2, \ldots, I_r\}$, where the intervals are ordered left-to-right (since all intervals in $\mathcal{I}_j$ are disjoint, this order is well-defined). For each $1 \leq x \leq r$, if x is odd, then we add the left endpoint of I_x to S_j and its right endpoint to T_j; otherwise, we add the right endpoint to S_j and the left endpoint to T_j. In the end, $|S_j| = |T_j| = |\mathcal{I}_j|$. This concludes the definition of the group-ICF instance. Let OPT be the value of the optimal solution to the MMIS instance (with no congestion), and let $\mathcal{I}^* = \bigcup_{j \in \mathcal{J}} \mathcal{I}_j^*$ be the optimal solution to the MMIS problem instance. Notice that for each job $j \in \mathcal{J}$, for each interval $I \in \mathcal{I}_j^*$, one of its endpoints is in S_j and the other is in T_j. For each interval $I \in \mathcal{I}^*$, we add the path connecting the endpoints of I to our solution. It is immediate to see that each group (S_i, T_i) is connected by OPT paths, and since intervals in $\mathcal{I}^*$ are disjoint, the congestion is bounded by 1.

Assume now that we are given a solution $\mathcal{P}^* = \bigcup_{j=1}^{n} \mathcal{P}_j^*$ to the group-ICF instance, where $\mathcal{P}_j^*$ is the set of paths connecting S_j to T_j, such that $|\mathcal{P}_j^*| \geq D'$, and the paths in $\mathcal{P}^*$ cause congestion at most c. We now transform $\mathcal{P}^*$ into a solution of value D' and congestion c for the original MMIS instance, as follows.

Consider any path P, whose endpoints are x, y, where one of these two vertices belongs to S_j and the other to T_j. We say that P is *canonical* iff x, y are endpoints of some interval $I \in \mathcal{I}_j$. If some path $P \in \mathcal{P}_j^*$ is not canonical, we can transform it into a canonical path, without increasing the overall congestion, as follows. We claim that path P must contain some interval $I \in \mathcal{I}_j$. Indeed, otherwise, let $\mathcal{S}_j$ be the set of endpoints of the intervals in $\mathcal{I}_j$. Then x and y are two consecutive vertices in $\mathcal{S}_j$, they are not endpoints of the same interval in $\mathcal{I}_j$, and yet one of them belongs to S_j and the other to T_j. But the sets S_j and T_j were defined in a way that makes this impossible. Therefore, P contains some interval $I \in \mathcal{I}_j$. We truncate path P so that it starts at the left endpoint of I and ends at the right endpoint of I. Once we process all paths in $\mathcal{P}^*$, we obtain a solution to the group-ICF problem instance, where all paths are canonical and the congestion is still bounded by c. This solution immediately gives us a solution of value D' and congestion at most c to the MMIS problem instance.

G.2 Proof of Theorem 17

The proof is almost identical to the hardness of approximation proof for Machine Minimization Job Scheduling of [CN06]. The only difference is that we create more intervals for each job, to ensure that the value of the optimal solution in the YES-INSTANCE is D. For completeness, we go over the whole proof here. Our starting point is exactly the same as in [CN06]: it is the standard 2-prover verifier for the 3SAT(5) problem, whose properties are summarized below.

A 2-prover Verifier

We perform a reduction from the 3SAT(5) problem, in which we are given a 3SAT formula φ with $5n/3$ clauses, where each clause contains exactly three distinct literals, and each literal appears in exactly five different clauses. The following statement is equivalent to the PCP theorem [AS98, ALM+98].

Theorem 25 *There is a constant $\delta > 0$ such that it is NP-hard to distinguish between the formula φ that is satisfiable and the one for which any assignment satisfies at most $(1-\delta)$ fraction of the clauses.*

We say that φ is a YES-INSTANCE if it is satisfiable, and if no assignment satisfies a $(1-\delta)$-fraction of the clauses, we say that it is a NO-INSTANCE. We use a standard 2-prover protocol for 3SAT(5), with ℓ parallel repetitions. In this protocol, there is a verifier and two provers. Given the input 3SAT(5) formula φ, the verifier chooses ℓ clauses $C_1, \ldots, C_\ell$ of φ independently at random, and for each $i : 1 \leq i \leq \ell$, a random variable x_i in clause C_i is chosen. The verifier then sends one query to each one of the two provers. The query to the first prover consists of the indices of clauses $C_1, \ldots, C_\ell$, while the query to the second prover contains the indices of variables $x_1, \ldots, x_\ell$. The first prover is expected to return an assignment to all variables in the clauses $C_1, \ldots, C_\ell$, which must satisfy the clauses, and the second prover is expected to return an assignment to variables $x_1, \ldots, x_\ell$. Finally, the verifier accepts if and only if the answers from the two provers are consistent. Combining Theorem 25 with the parallel repetition theorem [Raz98], we obtain the following theorem:

Theorem 26 *There is a constant $\gamma > 0$ such that:*

- *If φ is a* YES-INSTANCE*, then there is a strategy of the two provers such that the acceptance probability of the verifier is* 1.
- *If φ is a* NO-INSTANCE*, then for any strategy of the provers, the acceptance probability of the verifier is at most $2^{-\gamma\ell}$.*

Let R be the set of all random strings of the verifier, $|R| = (5n)^\ell$. Denote the set of all possible queries to the first and second provers by $\mathcal{Q}_1$ and $\mathcal{Q}_2$ respectively, so $|\mathcal{Q}_1| = (5n/3)^\ell$ and $|\mathcal{Q}_2| = n^\ell$. For each query $q \in \mathcal{Q}_1 \cup \mathcal{Q}_2$, we denote by $A(q)$, the collection of all possible answers to q (if $q \in \mathcal{Q}_1$, then we only include answers that satisfy all clauses in q). Notice that for each $q \in \mathcal{Q}_1$, $|A(q)| = 7^\ell$ and for each $q \in \mathcal{Q}_2$,

$|A(q)| = 2^\ell$. Given a random string $r \in R$ chosen by the verifier, we let $q_1(r)$ and $q_2(r)$ denote the queries sent to the first and second provers respectively given r.

This protocol defines a set $\Phi \subseteq \mathcal{Q}_1 \times \mathcal{Q}_2$ of constraints, where $(q_1, q_2) \in \Phi$ if and only if for some random string $r \in R$, $q_1 = q_1(r)$ and $q_2 = q_2(r)$. For each constraint $(q_1, q_2) \in \Phi$, we have a corresponding projection $\pi_{q_1,q_2} : A(q_1) \to A(q_2)$, which specifies the pairs of consistent answers for constraint (q_1, q_2).

We define another set $\Psi \subseteq \mathcal{Q}_1 \times \mathcal{Q}_1$ of constraints as follows: $(q_1, q_1') \in \Psi$ if and only if there is a query $q_2 \in \mathcal{Q}_2$ such that $(q_1, q_2) \in \Phi$ and $(q_1', q_2) \in \Phi$. Given a constraint $(q_1, q_1') \in \Psi$, a pair of answers $a \in A(q_1), a' \in A(q_1')$ is a satisfying assignment to this constraint iff $\pi_{q_1,q_2}(a) = \pi_{q_1',q_2}(a')$.

The rest of the reduction consists of two steps. First, we define a basic instance of MMIS. We then define our final construction, by combining a number of such basic instances together.

The Basic Instance

In this section, we construct instances of MMIS that are called basic instances, and analyze their properties. A basic instance is determined by several parameters, and the final construction combines a number of basic instances with different parameters.

Let c and D be parameters that we set later. We set the parameter ℓ of the Raz verifier to be $\ell = 3c/\gamma$, where γ is the constant from Theorem 26. A basic instance is determined by the 3SAT(5) formula φ that we reduce from, and the following parameters:

- An integer k;
- For each $q \in \mathcal{Q}_1$, a collection of $k(q) \leq k$ subsets of assignments $\mathcal{A}_1^q, \ldots, \mathcal{A}_{k(q)}^q \subseteq A(q)$, where for each $1 \leq i \leq k(q)$, $|\mathcal{A}_i^q| \geq |A(q)| - c$.

In order to define the job intervals, we will first define special intervals that we call *virtual intervals*. Unlike job intervals, virtual intervals are not part of the problem input, but they will be useful in defining the actual job intervals. There are three types of virtual intervals.

1. For each query $q \in \mathcal{Q}_1$, we have a virtual interval representing q and denoted by $I(q)$. This interval is called a *query interval*. All query intervals are equal-sized and non-overlapping.
2. Each query interval $I(q)$ is subdivided into $k(q)$ equal-sized non-overlapping virtual intervals representing the subsets $\mathcal{A}_i^q$ for $1 \leq i \leq k(q)$. An interval corresponding to set $\mathcal{A}_i^q$ is denoted by $I(\mathcal{A}_i^q)$.
3. Finally, each interval $I(\mathcal{A}_i^q)$ is subdivided into $|\mathcal{A}_i^q|$ non-overlapping virtual intervals, where each such interval represents a distinct assignment $a \in \mathcal{A}_i^q$. We denote this interval by $I_i^q(a)$, and we call such intervals *assignment intervals*. Observe that the same assignment a may appear in several subsets $\mathcal{A}_i^q$, and will be represented by a distinct interval for each such subset.

The set $\mathcal{J}$ of jobs for the basic instance is defined as follows. For each constraint $(q, q') \in \Psi$, for each pair $i : 1 \leq i \leq k(q)$, $i' : 1 \leq i' \leq k(q')$ of indices, there is a job $j(\mathcal{A}_i^q, \mathcal{A}_{i'}^{q'})$ in $\mathcal{J}$ if and only if there is **no** pair a, a' of assignments such that $a \in A(q) \setminus \mathcal{A}_i^q, a' \in A(q') \setminus \mathcal{A}_{i'}^{q'}$, and (a, a') is a satisfying assignment for the constraint (q, q'). If the job $j = j(\mathcal{A}_i^q, \mathcal{A}_{i'}^{q'})$ exists, we define a set $\mathcal{I}_j$ of $D(|\mathcal{A}_i^q| + |\mathcal{A}_{i'}^{q'}|)$ intervals for j, as follows. For each assignment $a \in \mathcal{A}_i^q$, we construct D intervals, which are completely contained in $I_i^q(a)$, and add them to $\mathcal{I}_j$. Similarly, for each assignment $a' \in \mathcal{A}_{i'}^{q'}$, we construct D intervals that are completely contained in $I_{i'}^{q'}(a')$, and add them to $\mathcal{I}_j$.

Consider some query $q \in \mathcal{Q}_1$, index $1 \leq i \leq k(q)$, and assignment $a \in \mathcal{A}_i^q$. Let $\mathcal{I}'$ be the set of indices of all job intervals that need to be contained in $I_i^q(a)$. We then create $|\mathcal{I}'|$ equal-sized non-overlapping sub-intervals of $I_i^q(a)$, and they serve as the intervals corresponding to the indices in $\mathcal{I}'$. This finishes the definition of the basic instance. Notice that all intervals in set $\bigcup_{j \in \mathcal{J}} \mathcal{I}_j$ are mutually disjoint. We note that the basic instance is

defined exactly like in [CN06], except that we create D intervals for each job j inside each relevant assignment interval $I_i^q(a)$, while in [CN06] only one such interval is created. In other words, the construction in [CN06] is identical to our construction with $D = 1$.

In the next two lemmas, we summarize the properties of the basic instance. The lemmas and their proofs are mostly identical to [CN06].

Lemma 9 *Assume that φ is a* YES-INSTANCE. *For each $q \in \mathcal{Q}_1$, let $f(q)$ denote the assignment to q obtained from the strategy of the provers, for which the acceptance probability of the verifier is 1. Then we can select, for each job $j \in \mathcal{J}$, a subset $\mathcal{I}_j^* \subseteq \mathcal{I}_j$ of D intervals, such that only intervals of the form $I_i^q(a)$ for $a = f(q)$ are used in the resulting solution $\bigcup_{j \in \mathcal{J}} \mathcal{I}_j^*$.*

Proof: Consider some job $j = j(\mathcal{A}_i^q, \mathcal{A}_{i'}^{q'})$. Let $a = f(q)$ and $a' = f(q')$, so a and a' are consistent. Due to the way we define the jobs, either $a \in \mathcal{A}_i^q$ or $a' \in \mathcal{A}_{i'}^{q'}$. Assume w.l.o.g. that it is the former case. Then we let $\mathcal{I}_j^*$ be the set of D intervals of job j that are contained in $I_i^q(a)$. □

Let $\mathcal{I}^*$ be any solution to the basic instance. We say that the interval $I(\mathcal{A}_i^q)$ is *used* by $\mathcal{I}^*$, iff there is some job interval $I \in \mathcal{I}^*$, such that $I \subseteq I(\mathcal{A}_i^q)$. We say that a query $q \in \mathcal{Q}_1$ is *good* iff for **all** $i : 1 \leq i \leq k(q)$, interval $I(\mathcal{A}_i^q)$ is used by the solution. The following lemma is identical to Claim 4.6 in [CN06], and its proof follows from [CN06].

Lemma 10 *Assume that φ is a* NO-INSTANCE. *Then for any solution $\mathcal{I}^*$ that contains at least one interval for each job, at least half of the queries are good.*

Construction size: Recall that $|\mathcal{Q}_1| = (5n/3)^\ell$, and each query $q \in \mathcal{Q}_1$ has at most k subsets $\mathcal{A}_i^q$ of assignments. For each such query q, there are 15^ℓ possible queries $q' \in \mathcal{Q}_1$ such that $(q, q') \in \Psi$. Therefore, there are at most $(5n/3)^\ell \cdot k^2 \cdot 15^\ell$ jobs in the basic instance. For each job j, $|\mathcal{I}_j| \leq 2 \cdot 7^\ell \cdot D$, and so the total number of intervals in the basic instance is bounded by $2 \cdot (5n/3)^\ell \cdot k^2 \cdot 15^\ell \cdot 7^\ell \cdot D = n^\ell \cdot 2^{O(\ell)} \cdot k^2 \cdot D$.

Notice that each query interval $I(q)$ contains at most $k^2 \cdot 15^\ell \cdot 7^\ell \cdot D = k^2 \cdot 105^\ell \cdot D$ job intervals.

The Final Construction

We combine c copies of the basic instance together to obtain the final construction. For $1 \leq h \leq c$, we will refer to the jobs and the job intervals from the hth instance as layer-h jobs or intervals. We will use different sets of parameters for different copies of the basic instance, and we denote the parameter k for the hth instance by k_h.

We start by defining, for each query $q \in \mathcal{Q}_1$, a virtual interval $I(q)$ on the real line. All intervals $I(q)$ are disjoint and equal-sized. For each query $q \in \mathcal{Q}_1$, interval $I(q)$ is used as the virtual interval representing q in all c basic instances that we construct.

Our final instance consists of c layers, where each layer is just a basic instance with carefully selected parameters. We maintain the following invariant: let $I_i^q(a)$ be any assignment interval, representing some assignments a to query q, at layer h. Then for each layer $h' < h$, there is precisely one virtual interval $\tilde{I}_{i'}^q(a')$ that contains $I_i^q(a)$, and $I_i^q(a)$ is disjoint from all other assignment intervals at level h'. Moreover, $a \neq a'$ must hold.

Layer 1 is a basic instance with the parameters $k_1 = 1$ and $\mathcal{A}_1^q = A(q)$ for all $q \in \mathcal{Q}_1$.

Assume now that we have defined layers $1, \ldots, h-1$, and we now define layer h. In order to do so, we specify, for each query $q \in \mathcal{Q}_1$, a collection $\mathcal{A}_1^q, \ldots, \mathcal{A}_{k(q)}^q$ of subsets of assignments to q, and for each such subset $\mathcal{A}_i^q$, we define the corresponding virtual interval $I(\mathcal{A}_i^q)$. We then plug in the basic instance with these new parameters, and the pre-specified virtual intervals $I(\mathcal{A}_i^q)$, that becomes the level-h instance.

Specifically, for each query $q \in \mathcal{Q}_1$, for each job interval $I' \subseteq I(q)$ that belongs to layer $h-1$, we define a new set $\mathcal{A}(I') \subseteq A(q)$ of assignments. The virtual interval corresponding to $\mathcal{A}(I')$ is I' itself. The subset $\mathcal{A}(I')$ is defined as follows. Recall that for each layer $h' < h$, interval I' is completely contained in some assignment

interval $I_i^q(a)$ of level h'. Let $a_1, \ldots, a_{h-1} \in A(q)$ be the assignments corresponding to these assignment intervals, where $a_{h'}$ is the assignment at level h'. We define $\mathcal{A}(I') = A(q) \setminus \{a_1, \ldots, a_{h-1}\}$.

We have therefore defined, for each query $q \in \mathcal{Q}_1$, a collection $\mathcal{A}_1^q, \ldots, \mathcal{A}_{k(q)}^q$ of subsets of assignments to q, and for each such subset $\mathcal{A}_i^q$, we have defined a corresponding virtual interval $I(\mathcal{A}_i^q)$. We now complete the construction of layer h by building a basic instance with the above parameters, using the virtual intervals $I(\mathcal{A}_i^q)$. This completes the description of layer h. Notice that for each query $q \in \mathcal{Q}_1$, for each $1 \leq i \leq k(q)$, we have $|\mathcal{A}_i^q| \geq |A(q)| - c$ as required.

Let k_h be the maximum, over all queries $q \in \mathcal{Q}_1$ of $k(q)$. Notice that k_h is precisely the maximum number of job intervals contained in any query interval $I(q)$ at level $h-1$ (since for each such interval we define a subset $\mathcal{A}_i^q$). The number of such job intervals is bounded by $k_{h-1}^2 \cdot 105^\ell \cdot D$, and so $k_h \leq \left(D \cdot 105^\ell\right)^{2^{h+1}}$ for all h. In particular, $k_c \leq D^{2^{c+1}} \cdot 2^{O(\ell \cdot 2^c)} = D^{2^{c+1}} \cdot 2^{O(2^c)}$ by the choice of $\ell = 3c/\gamma$. The final number of job intervals is therefore bounded by

$$N \leq n^\ell \cdot 2^{O(\ell)} \cdot k_c^2 \cdot D \leq n^{O(c)} \cdot 2^{O(2^c)} \cdot D^{2^{c+2}}.$$

We denote by $\mathcal{J}$ the set of jobs in the final instance. Assume first that the parameter c is fixed. We then choose D to be large enough to ensure that $D^{2^{c+2}} \geq n^{O(c)} \cdot 2^{O(2^c)}$. It is easy to see that $D = \Omega\left(N^{1/2^{c+3}}\right)$.

We can fix c to be any integer between 1 and $O(\log \log n)$. This will ensure that $N = n^{O(\log \log n)}$. Finally, notice that $\log \log N = O(c + \log \log n)$, so c takes values between 1 and $O(\log \log N)$, as required.

We conclude our analysis with the following two lemmas, that are identical to the analysis in [CN06].

Lemma 11 *If φ is a* YES-INSTANCE*, then for each job $j \in \mathcal{J}$, there is a subset $\mathcal{I}_j^* \subseteq \mathcal{I}_j$ of D, such that $\mathcal{I}^* = \bigcup_j \mathcal{I}_j^*$ causes congestion* 1.

Proof: We apply Lemma 9 to each layer of the construction. For each job $j \in \mathcal{J}$, let $\mathcal{I}_j^*$ be the subset of D intervals of j returned by Lemma 9, and let $\mathcal{I}^* = \bigcup_j \mathcal{I}_j^*$. We claim that all intervals in $\mathcal{I}^*$ are mutually disjoint. Indeed, assume for contradiction that some pair $I, I' \in \mathcal{I}^*$ of intervals overlap. By construction, $I, I' \subseteq I(q)$ for some query $q \in \mathcal{Q}_1$, and they belong to different layers. Assume w.l.o.g. that I belongs to layer h, and let $I_i^q(a)$ be the assignment interval for layer h in which it is contained, and I' belongs to layer $h' < h$, and let $I_{i'}^q(a')$ be the assignment interval for layer h' in which I' is contained. Then by our construction $a \neq a'$ must hold. This contradicts Lemma 9, which ensures that $f(q) = a = a'$. □

Lemma 12 *If φ is a* NO-INSTANCE*, then any collection $\mathcal{I}^*$ of job intervals, that contains at least one interval for each job, causes congestion at least $c/2$.*

Proof: Let $\mathcal{I}^*$ be any collection of job intervals as above, and for each $1 \leq h \leq c$, let $\mathcal{I}_h^* \subseteq \mathcal{I}^*$ be the subset of intervals that lie in layer h. Recall that if φ is a NO-INSTANCE, then for each layer h, at least half of the queries are good for $\mathcal{I}_h^*$. Therefore, at least one query must be good for at least $c/2$ layers. Denote such query by q. We let $h_1, \ldots, h_{c/2}$ be indices of $c/2$ layers for which q is good. For each $x = 1, \ldots, c/2$, we select a job interval I_x at layer h_x as follows. Let I_1 be any job interval $I \subseteq I(q)$ that belongs to $\mathcal{I}_{h_1}^*$. Assume now that intervals $I_1, \ldots, I_{x-1}$ have been defined, and we define I_x. By our construction, layer h_x must contain some virtual interval $I(\mathcal{A}_{i'}^q)$, that is completely contained in I_{x-1}. Since query q is good at layer h_x, $I(\mathcal{A}_{i'}^q)$ is used by $\mathcal{I}_{h_x}^*$, so there is a job interval $I' \subseteq I(\mathcal{A}_{i'}^q)$ in $\mathcal{I}_x^*$. We choose $I_x = I'$.

We have therefore defined a collection $I_1, \ldots, I_{c/2}$ of job intervals, such that $I_1 \supseteq I_2 \supseteq \ldots \supseteq I_{c/2}$, and all these intervals belong to $\mathcal{I}^*$. Therefore, the congestion of $\mathcal{I}^*$ is at least $c/2$. □

H An Algorithm for group-ICF

We again start with a special case of the problem, where all edge capacities are unit, and all demands are uniform. By appropriately scaling the demands, we can then assume w.l.o.g. that $\lambda_{\text{OPT}} = 1$, and the demand for

each pair (S_i, T_i) is D. Let $m = \lceil 16 \log n \rceil$ be a parameter. We later define a parameter $\Delta = k \operatorname{poly} \log n$, and we assume throughout the algorithm that:

$$D \geq 640 \Delta m \alpha_{\text{EDP}} \log^2 n = k \operatorname{poly} \log n. \tag{6}$$

We say that a group-ICF instance $(G, \mathcal{D})$ is a *canonical instance* iff for all $1 \leq i \leq k$, $S_i = \left\{s_1^i, \ldots, s_{mD}^i\right\}$, $T_i = \left\{t_1^i, \ldots, t_{mD}^i\right\}$, and there is a set $\mathcal{P} = \left\{P_j^i \mid 1 \leq i \leq k, 1 \leq j \leq mD\right\}$ of paths, where path P_j^i connects s_j^i to t_j^i, and the paths in $\mathcal{P}$ cause congestion at most $2m$ in G. We denote by $\mathcal{M}_i = \left\{(s_j^i, t_j^i)\right\}_{j=1}^{mD}$ the set of pairs corresponding to (S_i, T_i) for each $1 \leq i \leq k$, and we associate a canonical fractional solution f^* with such an instance, where we send $1/(2m)$ flow units along each path P_j^i. The value of such a solution is $D/2$ - the total amount of flow sent between each pair (S_i, T_i).

We now show that any instance $(G, \mathcal{D})$ of group-ICF with unit edge capacities and uniform demands can be converted into a canonical instance, by losing a factor 2 in the fractional solution value.

Let $(G, \mathcal{D})$ be an instance of group-ICF with uniform demands and unit edge capacities, and let f be the optimal fractional solution, whose congestion is at most 1. For each $1 \leq i \leq k$, let f_i be the flow in f that originates at the vertices of S_i and terminates at the vertices of T_i. Recall that f_i sends D flow units from the vertices of S_i to the vertices of T_i.

For each $1 \leq i \leq k$, we select a set $\mathcal{P}_i = \left\{P_1^i, \ldots P_{mD}^i\right\}$ of mD flow-paths connecting the vertices in S_i to the vertices of T_i, as follows. Each path P_j^i, for $1 \leq j \leq mD$, is selected independently at random, from the set of paths carrying non-zero flow in f_i, where each path P is selected with probability $f_i(P)/D$, (here $f_i(P)$ is the amount of flow on path P). Once the set $\mathcal{P}_i$ of paths for each $1 \leq i \leq k$ is selected, we define a new fractional solution f', where for each $1 \leq i \leq k$, we send $\frac{1}{2m}$ flow units on each path in $\mathcal{P}_i$. It is easy to see that for each $1 \leq i \leq k$, the amount of flow routed between S_i and T_i is exactly $D/2$. Moreover, using the standard Chernoff bounds, it is easy to see that w.h.p. flow f' causes congestion at most 1 in G. From now on, we will assume w.l.o.g. that the input instance is a canonical one, and that the set $\mathcal{P} = \left\{P_j^i \mid 1 \leq i \leq k, 1 \leq j \leq mD\right\}$ of paths is given. We denote the corresponding canonical fractional solution, where each path in $\mathcal{P}$ carries $1/(2m)$ flow units by f^*. For each $1 \leq i \leq k$, we now have a set $\mathcal{M}_i = \left\{(s_j^i, t_j^i)\right\}_{j=1}^{mD}$ of new demand pairs, and for each pair $(s,t) \in \mathcal{M}_i$, there is a path in $\mathcal{P}$ connecting s to t. Let $\mathcal{M} = \bigcup_{i=1}^k \mathcal{M}_i$.

Throughout this section, the parameter L in the definition of small and critical clusters is set to $L = O(\log^{25} n)$, and we set the precise value of L later.

H.1 Split Instances and Good Q-J Decompositions

In this section we introduce two special cases of the group-ICF problem and show efficient algorithms for solving them. In the following section we show an algorithm for the general problem, which decomposes an input instance of group-ICF into several sub-instances, each of which belongs to one of the two the special cases described here.

Split Instances. The first special case that we define is a split instance. Suppose we are given a canonical instance $(G, \mathcal{D})$ of the group-ICF problem, with the corresponding set $\mathcal{P}$ of paths connecting the demand pairs in $\bigcup_{i=1}^k \mathcal{M}_i$. Assume further that we are given a collection $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ of disjoint vertex subsets of G, such that each path $P \in \mathcal{P}$ is completely contained in one of the sets C_h. For each $1 \leq i \leq k$, for each $1 \leq h \leq \ell$, let $\mathcal{P}_i(C_h) \subseteq \mathcal{P}_i$ be the subset of paths contained in C_h. We say that instance $(G, \mathcal{D})$ is a split instance iff for each $1 \leq i \leq k$, for each $1 \leq h \leq \ell$, $|\mathcal{P}_i(C_h)| \leq \frac{D}{4\alpha_{\text{EDP}} \cdot \log^2 n} = \frac{D}{\operatorname{poly} \log n}$.

Theorem 27 *Let $(G, \mathcal{D})$ be a canonical split instance as described above. Then there is an efficient randomized algorithm that finds a collection $\mathcal{R}$ of paths that cause a congestion of at most η_{EDP} in G, and for each $1 \leq i \leq k$, at least $\frac{D}{64\alpha_{\text{EDP}} \cdot \log n} = \frac{D}{\operatorname{poly} \log n}$ paths connect the vertices of S_i to the vertices of T_i in $\mathcal{R}$ w.h.p.*

Proof: For each $1 \leq h \leq \ell$, let $\mathcal{P}(C_h) \subseteq \mathcal{P}$ be the subset of paths contained in C_h, and let $\mathcal{M}(C_h) \subseteq \mathcal{M}$ be the set of pairs of terminals that these paths connect. Recall that the paths in set $\mathcal{P}(C_h)$ cause a congestion of at most $2m$ in graph $G[C_h]$. Therefore, if we route $1/(2m)$ flow units along each path $P \in \mathcal{P}(C_h)$, then we obtain a feasible fractional solution to the EDP instance on graph $G[C_h]$ and the set $\mathcal{M}(C_h)$ of demands, where the value of the solution is $\frac{|\mathcal{M}(C_h)|}{2m}$.

Let $N = 2m\alpha_{\text{EDP}} \cdot \ln n$. We partition the set $\mathcal{M}(C_h)$ into N subsets $\mathcal{M}_1(C_h), \ldots, \mathcal{M}_N(C_h)$, and for each $1 \leq z \leq N$, we find a collection $\mathcal{R}_z(C_h)$ of paths contained in graph $G[C_h]$ that connect the demands in $\mathcal{M}_z(C_h)$ and cause congestion at most η_{EDP}.

We start with the set $\mathcal{M}(C_h)$ of demands, and apply Theorem 5 to input $(G[C_h], \mathcal{M}(C_h))$. Since there is a fractional solution of value $\frac{|\mathcal{M}(C_h)|}{2m}$, we obtain a collection $\mathcal{R}_1(C_h)$ of paths connecting a subset $\mathcal{M}_1(C_h) \subseteq \mathcal{M}(C_h)$ of at least $\frac{|\mathcal{M}(C_h)|}{2m\alpha_{\text{EDP}}}$ demand pairs. We then remove the pairs in $\mathcal{M}_1(C_h)$ from $\mathcal{M}(C_h)$ and continue to the next iteration. Clearly, after N iterations, every pair in the original set $\mathcal{M}(C_h)$ belongs to one of the subsets $\mathcal{M}_1(C_h), \ldots, \mathcal{M}_N(C_h)$. For each such subset $\mathcal{M}_z(C_h)$, we have computed an integral routing $\mathcal{R}_z(C_h)$ of the pairs in $\mathcal{M}_z(C_h)$ inside $G[C_h]$, with congestion at most η_{EDP}. We now choose an index $z \in \{1, \ldots, N\}$ uniformly at random, and set $\mathcal{M}'(C_h) = \mathcal{M}_z(C_h)$, and $\mathcal{R}(C_h) = \mathcal{R}_z(C_h)$. We view $\mathcal{R}(C_h)$ as the final routing of pairs in $\mathcal{M}'(C_h)$ inside the cluster C_h, and we say that all pairs in $\mathcal{M}'(C_h)$ are routed by this solution. Notice that the probability that a pair in $\mathcal{M}(C_h)$ is routed is $1/N$. The output of the algorithm is $\mathcal{R} = \bigcup_{h=1}^{\ell} \mathcal{R}(C_h)$.

We say that a demand pair (S_i, T_i) is satisfied iff $\mathcal{R}$ contains at least $\frac{mD}{2N} = \frac{D}{4\alpha_{\text{EDP}} \cdot \log n}$ paths connecting the vertices of S_i to the vertices of T_i. We now show that w.h.p. every demand pair (S_i, T_i) is satisfied.

Indeed, consider some demand pair (S_i, T_i). Recall that $|\mathcal{P}_i| = Dm$, and for each cluster $C_h \in \mathcal{C}$, $\mathcal{P}_i(C_h) \subseteq \mathcal{P}_i$ is the subset of paths contained in C_h. Let $\mathcal{M}_i(C_h) \subseteq \mathcal{M}_i$ be the set of pairs of endpoints of paths in $\mathcal{P}_i(C_h)$. Denote $D^* = \frac{D}{64\alpha_{\text{EDP}} \log^2 n}$, and recall that we $|\mathcal{M}_i(C_h)| \leq D^*$ must hold. We now define a random variable $y_{i,h}$ as follows. Let $n_{i,h}$ be the number of pairs of vertices in $\mathcal{M}_i(C_h)$ that are routed by $\mathcal{R}(C_h)$. Then $y_{i,h} = \frac{n_{i,h}}{D^*}$. Observe that the variables $\{y_{i,h}\}_{h=1}^{\ell}$ are independent random variables that take values in $[0, 1]$. Let $Y_i = \sum_{h=1}^{\ell} y_{i,h}$. Then the expectation of Y_i is $\mu_i = \frac{mD}{D^* \cdot N} = \frac{64m\alpha_{\text{EDP}} \log^2 n}{2m\alpha_{\text{EDP}} \cdot \log n} = 32 \log n$.

The probability that (S_i, T_i) is not satisfied is the probability that $Y_i < \frac{mD}{2D^* \cdot N} = \mu_i/2$. By standard Chernoff bounds, this is bounded by $e^{-\mu_i/8} \leq e^{-4 \log n} = 1/n^4$. From the union bound, with probability at least $(1 - 1/n^3)$, all pairs (S_i, T_i) are satisfied. □

Good Q-J Decompositions. Suppose we are given a canonical instance $(G, \mathcal{D})$ of the group-ICF problem, and any valid Q-J decomposition $(\mathcal{Q}, \mathcal{J})$ of the graph G. Recall that for each critical cluster $Q \in \mathcal{Q}$, we are given a partition $\pi(Q)$ of Q into small clusters that have the bandwidth property. Let $\mathcal{X} = \mathcal{J} \cup \left(\bigcup_{Q \in \mathcal{Q}} \pi(Q) \right)$. We say that the Q-J decomposition $(\mathcal{Q}, \mathcal{J})$ is *good* iff $\mathcal{Q} \neq \emptyset$, and no demand pair (s_j^i, t_j^i) is contained in a single cluster in $\mathcal{X}$. In other words, for each $1 \leq i \leq k$, for each $1 \leq j \leq mD$, vertices s_j^i and t_j^i must belong to distinct clusters in $\mathcal{X}$. We show that if we are given a good Q-J decomposition for G, then we can efficiently find a good integral solution for it.

For technical reasons that will become apparent later, we state the next theorem for canonical instances with non-uniform demands. The proof of the next theorem is deferred to Section H.3.

Theorem 28 *Assume that we are given a graph G, and for $1 \leq i \leq k$, two collections $S_i = \left\{s_1^i, \ldots, s_{D_i}^i\right\}$, $T_i = \left\{t_1^i, \ldots, t_{D_i}^i\right\}$ of vertices, where $D_i = \Omega\left(L^2 \log^{11} n\right)$. Assume further that we are given a set $\mathcal{P} = \left\{P_j^i \mid 1 \leq i \leq k, 1 \leq j \leq D_i\right\}$ of paths, where path P_j^i connects s_j^i to t_j^i, and the paths in $\mathcal{P}$ cause congestion at most $2m$ in G. Assume also that we are given a good Q-J decomposition $(\mathcal{Q}, \mathcal{J})$ for G. Then there is an efficient algorithm that finds an integral solution to the group-ICF instance, whose congestion is at most $c_{\mathtt{good}}$, and for each $1 \leq i \leq k$, at least $\left\lfloor D_i/\alpha_{\mathtt{good}} \right\rfloor$ paths connect vertices of S_i to vertices of T_i. Here, $\alpha_{\mathtt{good}} = O(\log^{42} n \operatorname{poly} \log \log n)$, and $c_{\mathtt{good}}$ is a constant.*

H.2 The Algorithm

We now present an algorithm to solve a general canonical instance. Our algorithm uses the following parameter:

$$\Delta = \max\left\{640kmL\alpha_{\text{ARV}}(n)\log n, \alpha_{\text{EDP}}, 16mL\alpha_{\texttt{good}}c_{\texttt{good}}, \Omega(L^2\log^{11} n)\right\} = k\operatorname{poly}\log n. \tag{7}$$

Let $\mathcal{T} = \bigcup_{i=1}^{k}(S_i \cup T_i)$ be the set of all terminals. The main idea is that we would like to find a Q-J decomposition of the graph G, such that each small cluster $X \in \mathcal{X}$, where $\mathcal{X} = \mathcal{J} \cup \left(\bigcup_{Q\in\mathcal{Q}}\pi(Q)\right)$, contains at most $D/\operatorname{poly}\log n$ terminals. Suppose we can find such a decomposition. We then say that a path $P \in \mathcal{P}$ is of type 1 iff its both endpoints are contained in some set $X \in \mathcal{X}$, and it is of type 2 otherwise. We can then partition the demand pairs (S_i, T_i) into two types: a demand pair (S_i, T_i) is of type 1 if most of the paths in $\mathcal{P}_i$ are of type 1, and it is of type 2 otherwise. This partitions the problem instance into two sub-instances: one induced by type-1 demand pairs, and the other induced by type-2 demand pairs. The former is a split instance w.r.t. $\mathcal{X}$, while for the latter instance, the current Q-J decomposition is a good decomposition. We can then solve the two resulting sub-instances using Theorems 27 and 28, respectively.

We are however unable to find such a Q-J decomposition directly. Instead, our algorithm consists of three steps. In the first step, we find a partition of $V(G)$ into subsets $V_1, \ldots, V_r$, and for each $1 \le h \le r$, we let $G_h = G[V_h]$. This partition guarantees that for each resulting graph G_h, for each small cut (A, B) in graph G_h, either A or B contain at most Δ terminals. Moreover, the number of edges $e \in E(G)$ whose endpoints do not lie in the same set V_h is at most $D/4$. This partition decomposes our original problem into r sub-problems, where for $1 \le h \le r$, the hth subproblem is defined over the graph G_h. In the second step, for each graph G_h, we find a Q-J decomposition $(\mathcal{Q}_h, \mathcal{J}_h)$. Let $\mathcal{X}_h = \mathcal{J} \cup \{\pi(Q) \mid Q \in \mathcal{Q}_h\}$ be the corresponding set of small clusters. While the Q-J decomposition is not necessarily good, we ensure that each cluster $C \in \mathcal{X}_h$ may only contain a small number of pairs (s_j^i, t_j^i) for all $1 \le i \le k$. We then decompose the demand pairs (S_i, T_i) into several types, and define a separate problem sub-instance for each of these types. We will ensure that each one of the resulting instances is either a split instance, or the Q-J decomposition computed in step 2 is a good decomposition for it.

Step 1: Partitioning the graph G. This step is summarized in the following lemma.

Lemma 13 *Let $(G, \mathcal{D})$ be a canonical instance of* group-ICF *with uniform demands. Then there is an efficient algorithm to find a partition $\mathcal{R}$ of V, such that for each $R \in \mathcal{R}$, for any partition (A, B) of R, where $|E_{G[R]}(A, B)| < 2L$, either A or B contain at most Δ terminals. Moreover, $\sum_{R\in\mathcal{R}} |\operatorname{out}(R)| \le D/4$.*

Proof: The proof is similar to standard well-linked decomposition procedures, except that we use slightly different parameters. Throughout the algorithm, we maintain a partition $\mathcal{R}$ of V, where at the beginning, $\mathcal{R} = \{V\}$.

Consider some set $R \in \mathcal{R}$. Let $G' = G[R]$, and let $\mathcal{T}_R = \mathcal{T} \cap R$. Let (A, B) be any partition of R, denote $\mathcal{T}_A = A \cap \mathcal{T}$, $\mathcal{T}_B = B \cap \mathcal{T}$, and assume w.l.o.g. that $|\mathcal{T}_A| \le |\mathcal{T}_B|$. We say that the cut (A, B) is *sparse* iff $|E_{G'}(A, B)| \le \frac{2L\cdot\alpha_{\text{ARV}}(n)}{\Delta} \cdot |\mathcal{T}_A|$. We run algorithm $\mathcal{A}_{\text{ARV}}$ on instance $(G', \mathcal{T}_R)$ of the sparsest cut problem. Let (A, B) be the output of this algorithm. If the cut (A, B) is sparse, then we remove R from $\mathcal{R}$, and we add A and B instead. Assume w.l.o.g. that $|\mathcal{T}_A| \le |\mathcal{T}_B|$. Then we charge the edges in $\mathcal{T}_A$ evenly for the edges in $|E_{G'}(A, B)|$. Notice that the charge to every edge is at most $\frac{2L\cdot\alpha_{\text{ARV}}(n)}{\Delta}$.

If algorithm $\mathcal{A}_{\text{ARV}}$ returns, for each $R \in \mathcal{R}$, a cut that is not sparse, then we stop the execution of the algorithm, and return the set $\mathcal{R}$. Notice that we are now guaranteed that for each set $R \in \mathcal{R}$, if (A, B) is any partition of R, $\mathcal{T}_A = A \cap \mathcal{T}$, $\mathcal{T}_B = B \cap \mathcal{T}$, and $|\mathcal{T}_A| \le |\mathcal{T}_B|$, then $|E_{G[R]}(A, B)| \ge \frac{2L}{\Delta} \cdot |\mathcal{T}_A|$. In particular, if $|E_{G[R]}(A, B)| < 2L$, then $|\mathcal{T}_A| < \Delta$ must hold.

In order to bound $\sum_{R\in\mathcal{R}} |\operatorname{out}(R)|$, consider some edge $e = (u, v)$. In each iteration that e is charged via the vertex u, the charge is at most $\frac{2L\alpha_{\text{ARV}}(n)}{\Delta}$, and the number of terminals in the cluster to which e belongs goes down by the factor of at least 2. Therefore, the total direct charge to e via u is at most $\frac{2L\alpha_{\text{ARV}}(n)\log n}{\Delta}$,

and the total direct charge to e via both is endpoints is bounded by $\frac{4L\alpha_{\text{ARV}}(n)\log n}{\Delta} \leq \frac{1}{32km}$ since $\Delta \geq 128kmL\alpha_{\text{ARV}}(n)\log n$. The indirect charge forms a geometric series, so overall we can bound: $\sum_{R\in\mathcal{R}} |\operatorname{out}(R)| \leq \frac{|\mathcal{T}|}{8km} = \frac{2kDm}{8km} = \frac{D}{4}$. □

We apply Lemma 13 to our instance $(G,\mathcal{D})$, to obtain a partition $\mathcal{R} = \{V_1,\ldots,V_r\}$ of $V(G)$. We denote $E' = \bigcup_{R\in\mathcal{R}} \operatorname{out}(R)$, and for each $1 \leq h \leq r$, we denote $G_h = G[V_h]$. Consider now some demand pair (S_i,T_i). Since $|E'| \leq D/4$, and the set $\mathcal{P}_i$ of paths causes congestion at most $2m$ in G, there are at least $mD/2$ pairs $(s_j^i,t_j^i) \in \mathcal{M}_i$, for which the path P_i^j is completely contained in some graph G_h. Let $\mathcal{M}_i' \subseteq \mathcal{M}_i$ be the subset of all such pairs (s_j^i,t_j^i), $|\mathcal{M}_i'| \geq mD/2$, and let $\mathcal{P}_i' \subseteq \mathcal{P}_i$ be the subset of their corresponding paths.

For each $1 \leq h \leq r$, let $\mathcal{M}_{i,h} \subseteq \mathcal{M}_i'$ be the subset of pairs (s_j^i,t_j^i) for which P_i^j is contained in G_h, let $\mathcal{P}_{i,h} \subseteq \mathcal{P}_i'$ be the subset of paths connecting such pairs, and let $D_{i,h} = |\mathcal{P}_{i,h}|$. Notice that $\sum_{h=1}^{r} |\mathcal{P}_{i,h}| \geq Dm/2$. For each $1 \leq h \leq r$, let $\mathcal{P}^h = \bigcup_{i=1}^{k} \mathcal{P}_{i,h}$, and let f^h be the fractional solution associated with the set $\mathcal{P}^h$ of paths, where each path in $\mathcal{P}^h$ is assigned $1/(2m)$ flow units. Then for each $1 \leq i \leq k$, flow f^h routes $D_{i,h}/(2m)$ flow units between the pairs in $\mathcal{M}_{i,h}$, and the flow f^h causes no congestion in G^h. We let $\mathcal{T}^h$ be the set of all terminals participating in pairs in $\bigcup_{i=1}^{k} \mathcal{M}_{i,h}$.

Step 2: constructing Q-J decompositions. We say that a graph G_h is small iff $|\mathcal{T}^h| \leq 8\Delta$, and otherwise we say that it is large. The goal of this step is to find a Q-J decomposition for each large graph G_h. In this step we fix some graph $G' = G_h$, where G_h is a large graph, and we focus on finding a Q-J decomposition for it. We denote $\mathcal{T}' = \mathcal{T}^h$ to simplify notation. Recall that $|\mathcal{T}'| > 8\Delta$.

Suppose we are given a Q-J decomposition $(\mathcal{Q},\mathcal{J})$ of G', and let $\mathcal{X} = \mathcal{J} \cup \left(\bigcup_{C\in\mathcal{Q}} \pi(C)\right)$. We say that this decomposition is *non-trivial* iff $\mathcal{Q} \neq \emptyset$. We say that it is *successful* iff each cluster $X \in \mathcal{X}$ contains at most Δ terminals in $\mathcal{T}'$. Notice that in general, Lemma 13 ensures that every cluster $X \in \mathcal{X}$ contains either at most Δ or at least $|\mathcal{T}'| - \Delta$ terminals from $\mathcal{T}'$, since for each $X \in \mathcal{X}$, $|\operatorname{out}_{G'}(X)| \leq L$. In order for a decomposition to be successful, we need to ensure that the latter case never happens.

We will instead achieve a decomposition with slightly weaker properties. Let $S^* \subseteq V(G')$ be any vertex subset with $|\operatorname{out}_{G'}(S^*)| \leq 2L$ (we do not require that $G'[S^*]$ is connected). Let G_{S^*} be the graph obtained from G' as follows: if $|\operatorname{out}_{G'}(S^*)| \leq L$, then we set $G_{S^*} = G' \setminus S^*$. Otherwise, $L < |\operatorname{out}_{G'}(S^*)| \leq 2L$, and we obtain G_{S^*} from G' by contracting the vertices of S^* into a super-node v_{S^*}. In this step, we show an efficient algorithm to find a set S^* with $|\operatorname{out}_{G'}(S^*)| \leq 2L$, together with a successful non-trivial Q-J decomposition for the graph G_{S^*}. The algorithm is summarized in the next theorem.

Theorem 29 *There is an efficient algorithm to find a cluster $S^* \subseteq V(G')$ with $|\operatorname{out}_{G'}(S^*)| \leq 2L$, and a successful non-trivial Q-J decomposition for the graph G_{S^*}.*

Proof: Our first step is to find an initial critical cluster Q_0 in graph G', that will be used to initialize the set $\mathcal{Q}_0$, that we use as input to Theorem 10 in order to find a non-trivial Q-J decomposition. We start by finding some large cluster S in graph G' that has the bandwidth property and then use Lemma 1 to find a critical cluster $Q_0 \subseteq S$. The next claim is formulated for a slightly more general setting, in which we will use it later.

Claim 6 *Let $\mathbf{G}$ be any graph, and $\tilde{\mathcal{T}}$ be any set of terminals in $\mathbf{G}$ with $|\tilde{\mathcal{T}}| \geq 4\Delta$, such that for any partition (A,B) of the vertices of $\mathbf{G}$ with $|E_{\mathbf{G}}(A,B)| \leq L$, either A or B contain at most Δ terminals. Then there is an efficient algorithm to find a large cluster S in $\mathbf{G}$ that has the bandwidth property.*

Proof:

We say that a cut (A,B) of $V(\mathbf{G})$ is *balanced*, iff $|\tilde{\mathcal{T}} \cap A|, |\tilde{\mathcal{T}} \cap B| > \Delta$. We start with some arbitrary balanced cut (A,B), and assume w.l.o.g. that $|\tilde{\mathcal{T}} \cap A| \leq |\tilde{\mathcal{T}} \cap B|$. We then perform a number of iterations. In each iteration, we start with a balanced cut (A,B), where $|\tilde{\mathcal{T}} \cap A| \leq |\tilde{\mathcal{T}} \cap B|$. At the end of the iteration, we either declare that B is a large cluster with the bandwidth property, or find another balanced cut (A',B') with $|E_{\mathbf{G}}(A',B')| < |E_{\mathbf{G}}(A,B)|$. Therefore, after $|E(\mathbf{G})|$ iterations, the algorithm must terminate with a large cluster that has the bandwidth property.

We now describe an execution of an iteration. We start with a balanced cut (A, B). Denote $\mathcal{T}_A = \tilde{\mathcal{T}} \cap A$ and $\mathcal{T}_B = \tilde{\mathcal{T}} \cap B$, and recall that $|\mathcal{T}_A| \leq |\mathcal{T}_B|$. Since the cut is balanced, and $|\mathcal{T}_A|, |\mathcal{T}_B| > \Delta$, $|E_{\mathbf{G}}(A, B)| \geq L$ must hold, so B is a large cluster.

We next run the algorithm $\mathcal{A}_{\text{ARV}}$ on the instance of the sparsest cut problem defined by the graph $\mathbf{G}[B]$, where the set of the terminals is the edges of $\text{out}_{\mathbf{G}}(B)$. If the output is a cut whose sparsity is greater than $\frac{1}{2}$, then we are guaranteed that B has the bandwidth property, so we output B. Assume now that the algorithm produces a cut (X, Y) whose sparsity is less than $\frac{1}{2}$. Let $\mathcal{T}_X = \tilde{\mathcal{T}} \cap X$ and $\mathcal{T}_Y = \tilde{\mathcal{T}} \cap Y$, and assume w.l.o.g. that $|\tilde{\mathcal{T}}_X| \leq |\tilde{\mathcal{T}}_Y|$. Let (A', B') be a new partition of $V(\mathbf{G})$, where $A' = A \cup X$, and $B' = Y$. It is easy to see that (A', B') remains a balanced cut, since $|\tilde{\mathcal{T}}| \geq 4\Delta$. It now only remains to show that $|E_{\mathbf{G}}(A', B')| < |E_{\mathbf{G}}(A, B)|$.

Indeed,

$$\begin{aligned}|E_{\mathbf{G}}(A', B')| &= |E_{\mathbf{G}}(X, Y)| + |E_{\mathbf{G}}(A, Y)| \\ &< \frac{|\text{out}_{\mathbf{G}}(X) \cap \text{out}_{\mathbf{G}}(B)|}{2} + |E_{\mathbf{G}}(A, Y)| < |E_{\mathbf{G}}(A, X)| + |E_{\mathbf{G}}(A, Y)| = |E_{\mathbf{G}}(A, B)|\end{aligned}$$

□

We can now use Lemma 1 to find a critical cluster $Q_0 \subseteq S$, where S is the large cluster returned by Claim 6. We set $\mathcal{Q}_0 = \{Q_0\}$, and use Theorem 10 to find a non-trivial Q-J decomposition $(\mathcal{Q}, \mathcal{J})$ of graph G' and set $\mathcal{T}'$ of terminals. If this decomposition is successful, then we are done. Assume therefore that some cluster $X \in \mathcal{X}$ contains at least $|\mathcal{T}'| - \Delta$ terminals of $\mathcal{T}'$. Notice that $|\text{out}_{G'}(X)| \leq L$ must hold, as X is a small cluster.

We now perform a number of iterations. In each iteration, we start with a cluster $S^* \subseteq V(G')$, with $|\text{out}_{G'}(S^*)| \leq 2L$, that contains at most Δ terminals of $\mathcal{T}'$ (for the first iteration, $S^* = V(G') \setminus X$). The output of the iteration is either a successful non-trivial Q-J decomposition of G_{S^*}, or a new cluster S' containing at most Δ terminals, with $|\text{out}_{G'}(S')| \leq 2L$, such that $|S'| > |S^*|$. In the latter case we replace S^* with S' and continue. Clearly, after $|V(G')|$ iterations, we will find the desired cluster S^* together with a successful Q-J decomposition for G_{S^*}.

We now describe an iteration. Let S^* be the current cluster, and assume first that $|\text{out}_{G'}(S^*)| \leq L$. Recall that in this case, $G_{S^*} = G' \setminus S^*$. Consider any partition (A, B) of G_{S^*}, such that $|E_{G_{S^*}}(A, B)| \leq L$, and assume w.l.o.g. that A contains fewer terminals than B. Then $|E_{G'}(A, B)| \leq 2L$, and so by the properties of Lemma 13, A contains at most Δ terminals. Therefore, for any partition (A, B) of the vertices of G_{S^*}, where $|E_{G_{S^*}}(A, B)| \leq L$, either A or B will contain at most Δ terminals. Moreover, $|\mathcal{T}' \setminus S^*| \geq 4\Delta$, since S^* contains at most Δ terminals and $|\mathcal{T}'| > 8\Delta$. We can now apply Claim 6 to graph $\mathbf{G} = G_{S^*}$ and a set $\tilde{\mathcal{T}} = \mathcal{T}' \setminus S^*$ of terminals, to find a large cluster S in G_{S^*}. We then find a critical cluster $Q_0 \subseteq S$ using Lemma 1, set $\mathcal{Q}_0 = \{Q_0\}$, and use Theorem 10 to find a non-trivial Q-J decomposition $(\mathcal{Q}, \mathcal{J})$ of G_{S^*}. If the decomposition is successful, then we are done. Otherwise, there is a cluster $X \in \mathcal{X}$ containing more than Δ terminals. Since $|\text{out}_{G_{S^*}}(X)| \leq L$, $|\text{out}_{G'}(X)| \leq 2L$ must hold, so from Lemma 13, X contains at least $|\mathcal{T}'| - \Delta$ terminals. Let $V' = V(G_{S^*}) \setminus X$. Then $|\text{out}_{G_{S^*}}(V')| \leq L$, so $|\text{out}_{G'}(V' \cup S^*)| \leq 2L$, and moreover $V' \cup S^*$ contain at most Δ terminals. We then update S^* to be $S^* \cup V'$, and continue to the next iteration.

Assume now that $|\text{out}_{G'}(S^*)| > L$. Recall that in this case G_{S^*} is obtained from G' by contracting the vertices of S^* into a super-node v_{S^*}. Since $|\text{out}_{G'}(S^*)| > L$, node v_{S^*} is a critical cluster for G_{S^*}. We then set $\mathcal{Q}_0 = \{\{v_{S^*}\}\}$, and apply Theorem 10 to compute a Q-J decomposition of G_{S^*}. If the decomposition is successful, then we are done. Otherwise, we will find a small cluster X in graph G_{S^*}, containing more than Δ terminals. Moreover, since $\{v_{S^*}\}$ is a critical cluster in $\mathcal{Q}$, vertex v_{S^*} does not belong to X. Therefore, X is also a small cluster in the graph G', and from Lemma 13, X contains at least $|\mathcal{T}'| - \Delta$ terminals. Let V' be the set of all vertices of graph G' that do not belong to X. Then $S^* \subseteq V'$, V' contains at most Δ terminals, and $|\text{out}_{G'}(V')| \leq L$. We are also guaranteed that $S^* \neq V'$, since $|\text{out}_{G'}(V')| \leq L$, while $|\text{out}_{G'}(S^*)| > L$. We now replace S^* with V' and continue to the next iteration. □

For each large graph G_h, let S^*_h be the set returned by Theorem 29, and let $(\mathcal{Q}_h, \mathcal{J}_h)$ be the Q-J decomposition of the corresponding graph that we denote by $G^h_{S^*}$. We denote $\mathcal{X}_h = \mathcal{J}_h \cup \left(\bigcup_{Q\in\mathcal{Q}_h} \pi(Q)\right)$, and $\mathcal{X}'_h = \mathcal{X}_h \cup \{S^*\}$. If G_h is small, then we denote $\mathcal{X}'_h = \{V(G_h)\}$. Finally, let $\mathcal{X} = \bigcup_{h=1}^{r} \mathcal{X}'_h$.

Consider some path $P \in \mathcal{P}'$. We say that this path is of type 1 iff it is completely contained in some cluster $X \in \mathcal{X}$. Assume now that the endpoints of P are contained in some cluster $X \in \mathcal{X}$, but P is not completely contained in cluster X. If $X = S^*_h$ for some $1 \le h \le r$, then we say that P is of type 2; otherwise, it is of type 3. All remaining paths are of type 4.

We partition the demand pairs (S_i, T_i) into four types. We say that a demand pair (S_i, T_i) is of type 1, iff at least $1/5$ of the paths in $\mathcal{P}'_i$ are of type 1; we say that it is of type 2 iff at least $1/5$ of the paths in $\mathcal{P}'_i$ are of type 2; similarly, it is of type 3 iff at least $1/5$ of the paths in $\mathcal{P}'_i$ are of type 3, and otherwise it is of type 4. If a pair belongs to several types, we select one of the types for it arbitrarily.

Step 3: Routing the demands We route the demands of each one of the four types separately.

Type-1 demands. It is easy to see that type-1 demands, together with the collection $\mathcal{X}$ of clusters, define a split instance. This is since each cluster $X \in \mathcal{X}$ contains at most Δ demand pairs, and $\Delta \le \frac{D}{640m\alpha_{\text{EDP}}\cdot\log^2 n}$ from Equation (6). If (S_i, T_i) is a type-1 demand, then the number of type-1 paths in $\mathcal{P}'_i$ is at least $Dm/10$. Therefore, we can apply Theorem 27, and obtain a collection $\mathcal{R}^1$ of paths that cause congestion at most η_{EDP} in G, and for each type-1 pair (S_i, T_i), at least $D/\operatorname{poly}\log n$ paths connect the vertices in S_i to the vertices in T_i in R^1 w.h.p.

Type-2 Demands We show that the set of type-2 demands, together with the collection of vertex subsets V_h where G_h is large, define a valid split instance. Indeed, for each such subset V_h of vertices, every type-2 path that is contained in V_h must contain an edge in $\operatorname{out}_{G_h}(S^*_h)$. Since there are at most $2L$ such edges, and the paths in $\mathcal{P}$ cause congestion at most $2m$, we get that the number of type-2 paths contained in each such subset V_h is bounded by $4mL < \Delta \le \frac{D}{640m\alpha_{\text{EDP}}\cdot\log^2 n}$. For each type-2 demand pair (S_i, T_i), there are at least $\frac{Dm}{10}$ type-2 paths connecting the vertices of S_i to the vertices of T_i in $\mathcal{P}'_i$. Therefore, we can apply Theorem 27, and obtain a collection $\mathcal{R}^2$ of paths that cause congestion at most η_{EDP} in G, and for each type-2 demand pair (S_i, T_i), at least $D/\operatorname{poly}\log n$ paths connect the vertices in S_i to the vertices in T_i in R^2 w.h.p.

Type-3 Demands Let $X \in \mathcal{X} \setminus \{S^*_1, \ldots, S^*_r\}$, and consider the set $\mathcal{P}(X)$ of type-3 paths whose both endpoints belong to X. Assume w.l.o.g. that $X \subseteq V_h$. Since $|\operatorname{out}_{G_h}(X)| \le 2L$, $|\mathcal{P}(X)| \le 4Lm$ must hold. Recall that $G_h[X]$ is a connected graph if $X \ne S^*_h$ for any $1 \le h \le r$. Let $\mathcal{M}(X)$ be the collection of pairs of endpoints of the paths in $\mathcal{P}(X)$. We select one pair $(s,t) \in \mathcal{M}(X)$ uniformly at random, and we connect s to t by any path contained in $G[X]$. Let $\mathcal{R}^3$ be the set of all such resulting paths. Using the same arguments as in Theorem 27, it is easy to see that w.h.p. every type-3 demand pair (S_i, T_i) has at least $D/\operatorname{poly}\log n$ paths connecting the vertices of S_i to the vertices of T_i in $\mathcal{R}^3$, since $4mL < \frac{D}{16\log^2 n}$.

Type-4 Demands Let (S_i, T_i) be any type-4 demand, and let $\mathcal{P}^4_i \subseteq \mathcal{P}'_i$ be the subset of type-4 paths for (S_i, T_i). Recall that $|\mathcal{P}^4_i| \ge Dm/5$. For each $1 \le h \le r$, let $\mathcal{P}^4_i(h) \subseteq \mathcal{P}^4_i$ be the subset of paths contained in the graph G_h. We say that pair (S_i, T_i) is *light* for G_h iff

$$|\mathcal{P}^4_i(h)| < \max\left\{16mL\alpha_{\mathtt{good}}c_{\mathtt{good}}, \Omega(L^2\log^{11} n)\right\}.$$

Otherwise, we say that it is *heavy* for G_h. We say that a demand pair (S_i, T_i) is light iff the total number of paths in sets $\mathcal{P}^4_i(h)$, where (S_i, T_i) is light for G_h is at least $Dm/10$. Otherwise, we say that it is heavy.

Let (S_i, T_i) be any demand pair, and let P be any type-4 path connecting a vertex of S_i to a vertex of T_i. Assume w.l.o.g. that P is contained in G_h for some $1 \le h \le r$. We say that P is a light path if (S_i, T_i) is light for G_h, and we say that it is a heavy path otherwise.

We now construct two canonical instances. The first instance consists of light (S_i, T_i) demand pairs of type 4, and their corresponding light type-4 paths in $\mathcal{P}'_i$. It is easy to see that this defines a split instance for the collection of vertex subsets V_h, where G_h is large. This is since for each light pair (S_i, T_i), for each subset

V_h where (S_i, T_i) is light for G_h, $|\mathcal{P}_i^4(h)| < \max\left\{16mL\alpha_{\mathtt{good}}c_{\mathtt{good}}, \Omega(L^2\log^{11} n\right\} \leq \Delta \leq \frac{D}{640m\alpha_{\text{EDP}}\cdot\log^2 n}$. Therefore, we can use Theorem 27 to find a collection $\mathcal{R}^4$ of paths that cause congestion at most η_{EDP}, and for each light type-4 pair (S_i, T_i), at least $D/\operatorname{poly}\log n$ paths connect the vertices of S_i to the vertices of T_i in $\mathcal{R}^4$ w.h.p.

Finally, consider some heavy type-4 pair (S_i, T_i). Let $\mathcal{P}_i'' \subseteq \mathcal{P}_i'$ be the set of all heavy type-4 paths in $\mathcal{P}_i'$, and let $\mathcal{M}_i''$ be the set of pairs of their endpoints. Recall that $|\mathcal{M}_i''| \geq Dm/10$, and the paths in $\mathcal{P}_i''$ cause congestion at most $2m$ in G. For each $1 \leq h \leq r$, where G_h is large, let $\mathcal{P}_i''(h) \subseteq \mathcal{P}_i''$ be the subset of paths contained in G_h, and let $\mathcal{M}_i''(h)$ be the set of their endpoints.

Consider some large graph G_h, and consider the sets (S_i', T_i') of demands for $1 \leq i \leq k$, where S_i' contains the first endpoint and T_i' contains the last endpoint of every path in $\mathcal{P}_i''(h)$. Then the Q-J decomposition that we have computed in Step 2 is a good decomposition for graph $G'_{S_h^*}$, where $G' = G_h$, for the set $(S_1', T_1'), \ldots, (S_k', T_k')$ of demands. Therefore, we can apply Theorem 28 to find a collection $\mathcal{R}^5(h)$ of paths in graph $G'_{S_h^*}$ that cause congestion at most $c_{\mathtt{good}}$ in G_h, and for each $1 \leq i \leq k$, at least $\lfloor \frac{|\mathcal{P}_i''(h)|}{2m\alpha_{\mathtt{good}}} \rfloor > 4Lc_{\mathtt{good}}$ paths connect vertices of S_i to vertices of T_i in $\mathcal{R}^5(h)$. Observe however that it is possible that $G'_{S_h^*}$ is obtained from G_h by contracting the vertices of S_h^* into a supernode v_h, and it is possible that some paths in $\mathcal{R}^5(h)$ contain the vertex v_h. However, since the degree of v_h is bounded by $2L$, and the congestion due to paths in $\mathcal{R}^5(h)$ is at most $c_{\mathtt{good}}$, there are at most $2Lc_{\mathtt{good}}$ such paths in $\mathcal{R}^5(h)$. We simply remove all such paths from $\mathcal{R}^5(h)$. Since for each $1 \leq i \leq k'$, $\mathcal{R}^5(h)$ contains more than $4Lc_{\mathtt{good}}$ paths connecting S_i to T_i, we delete at most half the paths connecting each pair (S_i, T_i) in set $\mathcal{R}^5(h)$

Let $\mathcal{R}^5 = \bigcup_h \mathcal{R}^5(h)$. Then for each heavy type-4 pair (S_i, T_i), at least $\frac{Dm}{40m\alpha_{\mathtt{good}}} = \frac{D}{\operatorname{poly}\log n}$ paths connect S_i to T_i in $\mathcal{R}^5$, since we are guaranteed that for all h, either $D_i(h) = 0$, or $D_i(h) \geq 2m\alpha_{\mathtt{good}}$. The congestion due to paths in $\mathcal{R}^5$ is bounded by $c_{\mathtt{good}}$.

Our final solution is $\mathcal{P}^* = \bigcup_{j=1}^5 \mathcal{R}^j$. From the above discussion, for every pair (S_i, T_i), set $\mathcal{P}^*$ contains at least $D/\operatorname{poly}\log n$ paths connecting S_i to T_i, and the congestion due to $\mathcal{P}^*$ is bounded by a constant.

It now only remains to prove Theorem 28. The extension of this algorithm to arbitrary edge capacities and demands appears in Section I.

H.3 Proof of Theorem 28

This section is dedicated to proving Theorem 28. Recall that we are given a canonical fractional solution, with a collection $\mathcal{P} = \left\{P_j^i : 1 \leq i \leq k, 1 \leq j \leq D_i\right\}$ of paths, such that path P_j^i connects s_j^i to t_j^i, and no path in $\mathcal{P}$ has its two endpoints in the same cluster $X \in \mathcal{X}$, where $\mathcal{X} = \mathcal{J} \cup \left(\bigcup_{Q\in\mathcal{Q}} \pi(Q)\right)$. We view each path $P_j^i \in \mathcal{P}$ as starting at vertex s_j^i and terminating at t_j^i. We will repeatedly use the following claim, whose proof follows from standard Chernoff bound, similarly to the proof of Theorem 22.

Claim 7 *Let $\mathcal{P}'$ be any collection of paths in G, and let $(\mathcal{P}_1', \ldots, \mathcal{P}_k')$ be any partition of $\mathcal{P}'$. Assume that we are given another partition $\mathcal{G}$ of the set $\mathcal{P}'$ of paths into groups of size at most q each, and assume further that for each $1 \leq i \leq k$, $|\mathcal{P}_i'| \geq 32q\log n$. Let $\mathcal{P}'' \subseteq \mathcal{P}'$ be a subset of paths, obtained by independently selecting, for each group $U \in \mathcal{G}$, a path $P_U \in U$ uniformly at random, so $\mathcal{P}'' = \{P_U \mid U \in \mathcal{G}\}$. Then for each $1 \leq i \leq k$, $|\mathcal{P}_i' \cap \mathcal{P}''| \geq \frac{|\mathcal{P}_i'|}{2q}$ with high probability.*

We say that a path $P \in \mathcal{P}$ is *critical* iff it is contained in some critical cluster $Q \in \mathcal{Q}$. We say that a pair (S_i, T_i) is critical iff the number of paths in $\mathcal{P}_i$ that are critical is at least $|\mathcal{P}_i|/2$. Otherwise, we say that (S_i, T_i) is a *regular* pair. We first show how to route the critical pairs, and then show an algorithm for routing regular pairs.

Routing Critical Pairs

Let $\tilde{\mathcal{P}} \subseteq \mathcal{P}$ be the set of all critical paths, and let $\mathcal{X}^Q = \bigcup_{Q\in\mathcal{Q}} \pi(Q)$. For each cluster $X \in \mathcal{X}^Q$, we define two sets of paths: $U_1(X)$, containing all paths in $\tilde{\mathcal{P}}$ whose first endpoint belongs to X, and $U_2(X)$, containing all

paths in $\tilde{\mathcal{P}}$ whose last endpoint belongs to X. Since cluster X cannot contain both endpoints of any path in $\tilde{\mathcal{P}}$, and the paths in $\tilde{\mathcal{P}}$ cause congestion at most $2m$, while $|\operatorname{out}(X)| \leq L$, we have that $|U_1(X)|, |U_2(X)| \leq 2mL$ for all $X \in \mathcal{X}^Q$. For each cluster $X \in \mathcal{X}^Q$, we then select, uniformly independently at random, one path $P_1(X) \in U_1(X)$, and one path $P_2(X) \in U_2(X)$. We then let $\tilde{\mathcal{P}}' \subseteq \tilde{\mathcal{P}}$ be the subset of paths P that were selected twice in this process. That is, $\tilde{\mathcal{P}}' = \left\{P_1(X) \mid X \in \mathcal{X}^Q\right\} \cap \left\{P_2(X) \mid X \in \mathcal{X}^Q\right\}$. Notice that both $\{U_1(X)\}_{X\in\mathcal{X}^Q}$ and $\{U_2(X)\}_{X\in\mathcal{X}^Q}$ are partitions of $\tilde{\mathcal{P}}$ into subsets of size at most $2mL$. Therefore, from Claim 7, for each $1 \leq i \leq k$, $|\mathcal{P}_i \cap \tilde{\mathcal{P}}'| \geq \frac{|\mathcal{P}_i|}{8m^2L^2}$ w.h.p.

We now fix some critical cluster $Q \in \mathcal{Q}$. Let $\mathcal{P}_Q \subseteq \tilde{\mathcal{P}}'$ denote the subset of paths in $\tilde{\mathcal{P}}'$ that are contained in Q, let $\mathcal{M}_Q$ be the set of pairs of their endpoints, and $\mathcal{T}_Q$ the subset of terminals that serve as endpoints to the paths in $\mathcal{P}_Q$. Recall that each small cluster $C \in \pi(Q)$ contains at most two terminals from $\mathcal{T}'$. We augment the graph G, by adding, for every terminal $t \in \mathcal{T}_Q$, an edge e_t connecting t to a new vertex t'. Let G_Q denote this new graph (but note that the new vertices t' do not belong to Q). We now show that Q still has the weight property in this new graph, and each cluster $C \in \pi(Q)$ still has the bandwidth property (with slighter weaker parameters). We can then use Theorem 9 for routing on critical clusters, to route the pairs in $\mathcal{M}_Q$.

Claim 8 *Each cluster $C \in \pi(Q)$ is $(\alpha_S/3)$-well-linked in graph G_Q, and $(Q, \pi(Q))$ has the weight property with parameter $\lambda' = \lambda/3$.*

Proof: Consider any small cluster $C \in \pi(Q)$. Cluster C contains at most two endpoints of paths in $\mathcal{P}_Q$, so for any subset $A \subseteq C$, $|\operatorname{out}_G(A)| \leq |\operatorname{out}_{G_Q}(A)| \leq |\operatorname{out}_G(A)| + 2$. Since C is α_S-well-linked in graph G, it is immediate that C is $\alpha_S/3$-well-linked in graph G_Q.

Consider now the graph H_Q, where we start with $G_Q[Q]$, and contract each cluster $C \in \pi$ into a super-node v_C, whose original weight $w(v_C)$ is $|\operatorname{out}_G(C)|$, and the new weight $w'(v_C)$ is $|\operatorname{out}_{G_Q}(C)|$. Notice that $w(v_C) \leq w'(v_C) \leq w(v_C)+2 \leq 3w(v_C)$. Since $(Q, \pi(Q))$ has the weight property with parameter λ in graph G, for any partition (A, B) of $V(H_Q)$,

$$|E_{H_Q}(A,B)| \geq \lambda \min\left\{\sum_{v\in A} w(v), \sum_{v\in B} w(v)\right\} \geq \frac{\lambda}{3} \cdot \min\left\{\sum_{v\in A} w'(v), \sum_{v\in B} w'(v)\right\}.$$

Therefore, $(Q, \pi(Q))$ has the weight property with parameter $\lambda' = \lambda/3$. □

We can now use Theorem 9 to find a grouping $\mathcal{G}_Q$ of the terminals in $\mathcal{T}_Q$ into groups of size $O(Z) = O(\log^4 n)$, such that for any set D of $(1, \mathcal{G}_Q)$-restricted demands, there is an efficient randomized algorithm that w.h.p. routes D integrally in $G[Q]$ with constant congestion. Grouping $\mathcal{G}_Q$ defines two partitions $\mathcal{G}_Q^1, \mathcal{G}_Q^2$ of the paths in $\mathcal{P}_Q$, as follows: for each group $U \in \mathcal{G}_Q$, we have a subset $\mathcal{P}_1(U) \subseteq \mathcal{P}_Q$ of paths whose first endpoint belongs to U, and a subset $\mathcal{P}_2(U) \subseteq \mathcal{P}_Q$ of paths whose last endpoint belongs to U. We let $\mathcal{G}_Q^1 = \{\mathcal{P}_1(U) \mid U \in \mathcal{G}_Q\}$, and $\mathcal{G}_Q^2 = \{\mathcal{P}_2(U) \mid U \in \mathcal{G}_Q\}$. For each group $U \in \mathcal{G}$, we randomly sample one path in $\mathcal{P}_1(U)$ and one path in $\mathcal{P}_2(U)$. We let $\mathcal{P}'_Q$ be the set of all paths that have been selected twice, once via their first endpoint and once via their last endpoint, and we let $\tilde{\mathcal{P}}'' = \bigcup_{Q\in\mathcal{Q}} \mathcal{P}'_Q$. From Claim 7, for each critical (S_i, T_i) pair, $|\mathcal{P}_i \cap \tilde{\mathcal{P}}''| \geq \Omega\left(\frac{|\mathcal{P}_i|}{m^2L^2Z^2}\right) = \Omega\left(\frac{|\mathcal{P}_i|}{L^2\log^{10} n}\right)$ w.h.p. For each $Q \in \mathcal{Q}$, let $\mathcal{M}'_Q$ be the set of pairs of endpoints of paths in $\mathcal{P}'_Q$. Then $\mathcal{M}'_Q$ defines a set of $(2, \mathcal{G}_Q)$-restricted demands on the set $\mathcal{T}_Q$ of terminals, and from Theorem 9, there is a randomized algorithm to route these demands in $G[Q]$ with constant congestion.

Routing Regular Pairs

We use the graph H, given by Theorem 11. The algorithm consists of two steps. In the first step, we route some of the terminals to the boundaries of the critical clusters, and create what we call "fake terminals". This step is very similar to the algorithm of Andrews [And10]. In this way, we transform the problem of routing the original terminal pairs in graph G into a problem of routing the new fake terminal pairs in graph H. The second step is very similar to the proof of Theorem 13: we split graph H into $x = \operatorname{poly}\log n$ sub-graphs $H_1, \ldots, H_x$ using

standard edge sampling, and route a subset of the fake demand pairs in each graph H_j using the algorithm of Rao and Zhou [RZ10].

Since we now focus on regular pairs only, to simplify notation, we assume that we are given a collection $\{(S_1,T_1),\ldots,(S_k,T_k)\}$ of regular demand pairs, and a collection $\mathcal{P}=\left\{P_j^i : 1\leq i\leq k, 1\leq j\leq D_i'\right\}$ of paths, where $D_i'=D_i/2$, such that path P_j^i connects s_j^i to t_j^i. We assume that all paths P_j^i are regular. For each $1\leq i\leq k$, $S_i=\left\{s_1^i,\ldots,s_{D_i'}^i\right\}$, and $T_i=\left\{t_1^i,\ldots,t_{D_i'}^i\right\}$. Let $\mathcal{T}=\bigcup_{i=1}^k(S_i\cup T_i)$ be the set of all terminals, and for $1\leq i\leq k$, let $\mathcal{M}_i=\left\{(s_j^i,t_j^i)\right\}_{j=1}^{D_i'}$ be the set of the pairs of endpoints of paths in set $\mathcal{P}_i=\left\{P_1^i,\ldots,P_{D_i}^i\right\}$. Denote by $\mathcal{T}^J$ and $\mathcal{T}^Q$ the sets of all terminals contained in the J- and the Q-clusters respectively, that is, $\mathcal{T}^J=\mathcal{T}\cap\left(\bigcup_{C\in\mathcal{J}}C\right)$, and $\mathcal{T}^Q=\mathcal{T}\cap\left(\bigcup_{C\in\mathcal{Q}}C\right)$.

Step 1: Defining fake terminal pairs. The goal of this step is to define new pairs of terminals, that we call fake terminal pairs, in graph H, so that a good routing of the fake terminal pairs in graph H immediately translates into a good routing of the original pairs in graph G. This step largely follows the ideas of [And10]. Let $E^Q=\bigcup_{Q\in\mathcal{Q}}\mathrm{out}(Q)$. Our first step is to route all terminals in $\mathcal{T}$ to the edges of E^Q. This is done using the following two lemmas.

Lemma 14 *There is an efficient algorithm to find a partition $\mathcal{G}^J$ of the set $\mathcal{T}^J$ of terminals into groups of size at most $48m$, such that for any $(2,\mathcal{G}^J)$-restricted subset $\mathcal{T}'\subseteq\mathcal{T}^J$ of terminals, there is an efficient algorithm to find a set $\mathcal{P}_J:\mathcal{T}'\rightsquigarrow_{12}E^Q$ of paths in G.*

Proof: Let $E^J=\bigcup_{J\in\mathcal{J}}\mathrm{out}(J)$. Since no pair of endpoints of paths in $\mathcal{P}$ is contained in a single cluster $J\in\mathcal{J}$, there is a set $\mathcal{P}_1:\mathcal{T}^J\rightsquigarrow_{4m}E^J$ of paths, connecting every terminal $t\in\mathcal{T}^J$ to some edge in E^J. The collection $\mathcal{P}_1$ of paths is obtained as follows: let $t\in\mathcal{T}$ be the endpoint of some path $P\in\mathcal{P}$, and assume that $t\in C$, where C is a J-cluster. Let $e\in\mathrm{out}(C)$ be the first edge on path P that is not contained in C. We then add the segment of P between t and e to the set $\mathcal{P}_1$ of paths. This defines the set $\mathcal{P}_1:\mathcal{T}^J\rightsquigarrow_{4m}E^J$ of paths.

Recall that we are also given a set N of paths called tendrils, $N:E^J\rightsquigarrow_3E^Q$. Combining the two sets of paths, we obtain a collection $\mathcal{P}_2:\mathcal{T}^J\rightsquigarrow_{16m}E^Q$.

We now use the standard grouping technique, to partition the terminals in $\mathcal{T}^J$ into groups of size at least $16m$ and at most $48m$. Let $\mathcal{G}^J$ be this resulting grouping. Since each group can simultaneously send one flow unit to the edges of E^Q with congestion 1, it is immediate to see that for any $(1,\mathcal{G}^J)$-restricted subset $\mathcal{T}'\subseteq\mathcal{T}^J$ of terminals, there is a flow $F_J:\mathcal{T}'\rightsquigarrow_2E^Q$. From the integrality of flow, there must be a set $\mathcal{P}_J':\mathcal{T}'\rightsquigarrow_2E^Q$ of paths, which can be computed efficiently. Finally, using Observation 1, we conclude that for any $(2,\mathcal{G}^J)$-restricted subset $\mathcal{T}'\subseteq\mathcal{T}^J$ of terminals, we can efficiently find a set $\mathcal{P}_J:\mathcal{T}'\rightsquigarrow_{12}E^Q$ of paths. □

Lemma 15 *We can efficiently find a partition $\mathcal{G}^Q$ of the set $\mathcal{T}^Q$ of terminals into groups of size at most $48m$, such that for any $(2,\mathcal{G}^Q)$-restricted subset $\mathcal{T}'\subseteq\mathcal{T}^Q$ of terminals, there is an efficient algorithm to find a set $\mathcal{P}_Q:\mathcal{T}'\rightsquigarrow_{12}E^Q$ of paths G.*

Proof: We start by showing that there is a collection $\mathcal{P}_2:\mathcal{T}^Q\rightsquigarrow_{2m}E^Q$ of paths in G, which is constructed as follows. Let $t\in\mathcal{T}$ be the endpoint of some path $P\in\mathcal{P}$, and assume that $t\in C$, where C is a Q-cluster. Since P is a regular path, it is not completely contained in C. Let $e\in\mathrm{out}(C)$ be the first edge on path P that is not contained in C. We then add the segment of P between t and e to the set $\mathcal{P}_2$ of paths. Since the paths in $\mathcal{P}$ cause congestion at most $2m$, the total congestion due to paths in $\mathcal{P}_2$ is bounded by $4m$. The rest of the proof is identical to the proof of Lemma 14: we compute a grouping $\mathcal{G}^Q$ of the terminals in $\mathcal{T}^Q$ into groups of size at least $4m$ and at most $12m$, using the standard grouping technique. It is then immediate to see that for any for any $(1,\mathcal{G}^Q)$-restricted subset $\mathcal{T}'\subseteq\mathcal{T}^Q$ of terminals, there is a flow $F_Q:\mathcal{T}'\rightsquigarrow_2E^Q$, and hence a set $\mathcal{P}_Q':\mathcal{T}'\rightsquigarrow_2E^Q$ of paths. Using Observation 1, we conclude that for any $(2,\mathcal{G}^Q)$-restricted subset $\mathcal{T}'\subseteq\mathcal{T}^Q$ of terminals, we can efficiently find a set $\mathcal{P}_Q:\mathcal{T}'\rightsquigarrow_{12}E^Q$ of paths. □

Let $\mathcal{G} = \mathcal{G}^Q \cup \mathcal{G}^J$ be the partition of terminals in $\mathcal{T}$ obtained from Lemmas 14 and 15. For each group $U \in \mathcal{G}$ of terminals, we define two subsets of paths: $\mathcal{P}_1(U) \subseteq \mathcal{P}$ contains all paths whose first endpoint belongs to U, and $\mathcal{P}_2(U) \subseteq \mathcal{P}$ contains all paths whose last endpoint belongs to U. We then select, independently uniformly at random, two paths $P_1(U) \in \mathcal{P}_1(U)$ and $P_2(U) \in \mathcal{P}_2(U)$. Let $\mathcal{P}' \subseteq \mathcal{P}$ be the subset of paths that have been selected twice, that is, $\mathcal{P}' = \{P_1(U) \mid U \in \mathcal{G}\} \cap \{P_2(U) \mid U \in \mathcal{G}\}$. Since both $\{\mathcal{P}_1(U)\}_{U\in\mathcal{G}}$ and $\{\mathcal{P}_2(U)\}_{U\in\mathcal{G}}$ define partitions of the paths in $\mathcal{P}$ into sets of size at most $48m$, from Claim 7, for each $1 \leq i \leq k$, $|\mathcal{P}_i \cap \mathcal{P}'| \geq \frac{|\mathcal{P}_i|}{4608m^2}$ w.h.p.

Let $\mathcal{T}' \subseteq \mathcal{T}$ be the set of terminals that serve as endpoints for paths in $\mathcal{P}'$. Since set $\mathcal{T}'$ is $(2, \mathcal{G})$-restricted, from Lemmas 14 and 15, there is a collection $\mathcal{R} : \mathcal{T}' \rightsquigarrow_{24} E^Q$ of paths in graph G. For each terminal $t \in \mathcal{T}'$, set $\mathcal{R}$ contains a path P_t, connecting t to some edge $e_t \in E^Q$. We define a mapping $f : \mathcal{T}' \to E^Q$, where $f(t) = e_t$.

Recall that for each critical cluster $Q \in \mathcal{Q}$, Theorem 9 gives a partition $\mathcal{G}(Q)$ of the edges of $\mathrm{out}(Q)$ into subsets of size at most $3Z = O(\log^4 n)$.

Consider some edge $e \in E^Q$. If there is a single critical cluster $Q \in \mathcal{Q}$ such that $e \in \mathrm{out}(Q)$, then we say that e belongs to Q. Otherwise, if there are two such clusters $Q_1, Q_2 \in \mathcal{Q}$, then we select one of them arbitrarily, say Q_1, and we say that e belongs to Q_1. We will view $\mathcal{G}(Q)$ as a partition of only those edges in $\mathrm{out}(Q)$ which belong to Q, and we will ignore all other edges. Let $\mathcal{G}' = \bigcup_{Q\in\mathcal{Q}} \mathcal{G}(Q)$, so $\mathcal{G}'$ is a partition of E^Q. Our final step is to sample the paths in $\mathcal{P}'$, such that for each group $U \in \mathcal{G}'$, there are at most two paths whose endpoints are mapped to the edges of U.

For each group $U \in \mathcal{G}'$, we define a subset $\mathcal{P}_1(U) \subseteq \mathcal{P}'$ of paths, containing all paths whose first endpoint t is mapped to an edge of U, that is, $f(t) \in U$, and similarly, a subset $\mathcal{P}_2(U) \subseteq \mathcal{P}'$ of paths, containing all paths whose last endpoint is mapped to an edge of U. We then select, uniformly independently at random, a path $P_1(U) \in \mathcal{P}_1(U)$, and a path $P_2(U) \in \mathcal{P}_2(U)$, and let $\mathcal{P}'' \subseteq \mathcal{P}'$ be the subset of paths that have been selected twice, that is, $\{P_1(U) \mid U \in \mathcal{G}\} \cap \{P_2(U) \mid U \in \mathcal{G}\}$. Since each set $|\mathcal{P}_1(U)|, |\mathcal{P}_2(U)| \leq 3Z = O(\log^4 n)$, from Claim 7, for each $1 \leq i \leq k$, $|\mathcal{P}_i \cap \mathcal{P}''| \geq \Omega\left(\frac{D_i}{m^2Z^2}\right) = \Omega\left(\frac{D_i}{\log^{10} n}\right)$ w.h.p.

Let $\mathcal{T}'' \subseteq \mathcal{T}$ be the set of terminals that serve as endpoints of paths in $\mathcal{P}''$, and let $\mathcal{R}'' \subseteq \mathcal{R}$ be their corresponding subset of paths, $\mathcal{R}'' : \mathcal{T}'' \rightsquigarrow_{24} E^Q$. For each $1 \leq i \leq k$, let $\mathcal{P}''_i = \mathcal{P}_i \cap \mathcal{P}''$, and let $\mathcal{M}''_i$ be the set of pairs of endpoints of the paths in $\mathcal{P}''_i$.

Let H be the graph given by Theorem 11. We are now ready to define fake demand pairs for the graph H. For each $1 \leq i \leq k$, we define a set $\tilde{\mathcal{M}}_i$ of demand pairs, and the sets $\tilde{S}_i, \tilde{T}_i$ of fake terminals, as follows: for each pair $(s,t) \in \mathcal{M}''_i$, if $Q_1 \in \mathcal{Q}$ is the critical cluster to which $f(s)$ belongs, and $Q_2 \in \mathcal{Q}$ the critical cluster to which $f(t)$ belongs, then we add (v_{Q_1}, v_{Q_2}) to $\tilde{\mathcal{M}}_i$, and we add v_{Q_1} to $\tilde{S}_i$ and v_{Q_2} to $\tilde{T}_i$. Notice that we allow $\tilde{\mathcal{M}}_i, \tilde{S}_i$ and $\tilde{T}_i$ to be multi-sets. This finishes the definition of the fake demand pairs. In order to complete Step 1, we show that any good integral solution to this new group-ICF instance will give a good integral solution to the original group-ICF instance.

Lemma 16 *Let $\tilde{\mathcal{P}}$ be any collection of paths in graph H that causes congestion at most γ. For each $1 \leq i \leq k$, let n_i be the number of paths in $\tilde{\mathcal{P}}$ connecting the fake terminals in $\tilde{S}_i$ to the fake terminals in $\tilde{T}_i$. Then we can efficiently find a collection $\mathcal{P}^*$ of paths in the original graph G, such that for each $1 \leq i \leq k$, there are n_i paths connecting the terminals of S_i to the terminals of T_i in $\mathcal{P}^*$, and the congestion caused by paths in $\mathcal{P}^*$ is at most $c_0\gamma$ for some constant c_0.*

Proof: Consider some path $\tilde{P} \in \tilde{\mathcal{P}}$, and let $(v_{Q_1}, v_{Q_2}) \in \tilde{\mathcal{M}}$ be the endpoints of path $\tilde{P}$. Let $(s,t) \in \mathcal{M}$ be the original pair of terminals that defined the pair (v_{Q_1}, v_{Q_2}) of fake terminals. We transform the path $\tilde{P}$ into a path P connecting s to t in graph G. In order to perform the transformation, we start with the path $\tilde{P}$, and replace its endpoints and edges with paths in graph G. Specifically, we replace v_{Q_1} with the path $P_s \in \mathcal{R}''$ that connects s to some edge in $\mathrm{out}(Q_1)$, and we replace v_{Q_2} with the path $P_t \in \mathcal{R}''$, connecting some edge in $\mathrm{out}(Q_2)$ to t. Additionally, for each edge $e = (v_Q, v_{Q'})$ on path P, we replace e by the path P_e connecting some edge in $\mathrm{out}(Q)$ to some edge in $\mathrm{out}(Q')$, given by Theorem 11. So far, each path $\tilde{P} \in \tilde{\mathcal{P}}$ is replaced by a sequence of

paths (that we call segments) in graph G. For each pair σ, σ' of consecutive segments, there is a critical cluster $Q \in \mathcal{Q}$, such that the last edge of σ and the first edge of σ' belong to $\mathrm{out}(Q)$. For each critical cluster $Q \in \mathcal{Q}$, we now define a set $\mathcal{D}(Q)$ of demands on the edges of $\mathrm{out}(Q)$, as follows: for each path $\tilde{P} \in \tilde{\mathcal{P}}$, for each pair (σ, σ') of consecutive segments that we have defined for path $\tilde{P}$, where the last edge e of σ, and the first edge e' of σ' belong to $\mathrm{out}(Q)$, we add the demand pair (e, e') to $\mathcal{D}(Q)$. We allow $\mathcal{D}(Q)$ to be a multi-set.

Since the paths in $\tilde{\mathcal{P}}$ cause congestion at most γ, from Property C4, and from the fact that the terminals in $\mathcal{T}''$ are $(2, \mathcal{G})$-restricted, we get that for each critical cluster $Q \in \mathcal{Q}$, the set $\mathcal{D}(Q)$ of demands is $(2\gamma+2)$-restricted. Combining Observation 1 with Theorem 9, we get that the set $\mathcal{D}(Q)$ of demands can be routed inside Q with congestion at most $547 \cdot (6\gamma+6)$. For each path $\tilde{P} \in \tilde{\mathcal{P}}$, we now combine the segments we have defined for $\tilde{P}$ with the routings we have computed inside the critical clusters to connect these segments, to obtain the final path P in graph G. □

Step 2: Routing in graph H. In this step, we find a solution for the group-ICF problem defined on graph H and the set of fake terminal pairs. For each $1 \leq i \leq k$, let $\tilde{D}_i = |\tilde{\mathcal{M}}_i|$, and recall that $\tilde{D}_i = \Omega\left(\frac{D_i}{\log^{10} n}\right)$. For each $1 \leq i \leq k$, we will route a polylogarithmic fraction of the demand pairs in $\tilde{\mathcal{M}}_i$, with no congestion in graph H. This step is almost identical to the proof of Theorem 13, except that we use different parameters. For simplicity, we assume w.l.o.g. in this step that all values $\tilde{D}_i$ are equal. In order to achieve this, let D be the minimum value of $\tilde{D}_i$ over all $1 \leq i \leq k$. For each $1 \leq i \leq k$, we partition the demand pairs in $\tilde{\mathcal{M}}_i$ into subsets, containing D pairs each (except possibly the last subset). If one of the resulting subsets contains fewer than D pairs, we simply disregard all pairs in this subset. In this way, we define a new collection of demand pairs $\left\{\tilde{\mathcal{M}}'_{i'}\right\}$, where $1 \leq i' \leq k'$. Notice that it is now enough to find a collection $\tilde{\mathcal{P}}$ of paths, such that for each new group $\tilde{\mathcal{M}}'_{i'}$, at least a poly-logarithmic fraction of the pairs in $\tilde{\mathcal{M}}'_{i'}$ are connected by paths in $\tilde{\mathcal{P}}$. To simplify notation, we now assume w.l.o.g. that for each $1 \leq i \leq k$, $\tilde{D}_i = D$.

Let $\alpha_{\text{RZ}} = O(\log^{10} n)$, $L_{\text{RZ}} = \Theta(\log^5 n)$ be the parameters from Theorem 24. Let $x = 16m\alpha_{\text{RZ}} \cdot \log n = O(\log^{12} n)$. We set $L = 2\alpha^* \cdot x \cdot L_{\text{RZ}} = O(\log^{25} n \operatorname{poly}\log\log n)$.

We split graph H into x graphs $H_1, \ldots, H_x$, as follows. For each $1 \leq j \leq x$, we have $V(H_j) = V(H)$. In order to define the edge sets of graphs H_j, each edge $e \in E$, chooses an index $1 \leq j \leq x$ independently uniformly at random, and is then added to $E(H_j)$. This completes the definition of the graphs H_j.

For convenience, we define a new graph G', which is obtained from the original graph G by contracting each cluster $Q \in \mathcal{Q}$ into a super-node v_Q. Notice that the set $\tilde{\mathcal{M}}$ of fake terminal pairs is also a set of pairs of vertices in graph G', so we can also view $\tilde{\mathcal{M}}$ as defining a set of demand pairs in graph G'.

Given any partition (A, B) of the vertices of $V(H)$, let $\mathrm{cut}_{G'}(A, B)$ denote the value of the minimum cut $|E_{G'}(A', B')|$ in graph G', such that $A \subseteq A', B \subseteq B'$. Theorem 11 guarantees that the size of the minimum cut in H is at least L/α^*, and for each partition (A, B) of $V(H)$, $\mathrm{cut}_{G'}(A, B) \leq \alpha^* \cdot |E_H(A, B)|$. From Theorem 12, for each graph H_j, for $1 \leq j \leq x$, w.h.p. we have that:

- The value of the minimum cut in H_j is at least $\frac{L}{2\alpha^* x} = L_{\text{RZ}}$.
- For any partition (A, B) of $V(H_j)$, $|E_{H_j}(A, B)| \geq \frac{\mathrm{cut}_{G'}(A,B)}{2x\alpha^*}$.

We need the following lemma.

Lemma 17 *For each $1 \leq j \leq x$, there is a fractional solution to the instance $(H_j, \tilde{\mathcal{M}})$ of* group-ICF*, where each demand pair in $\tilde{\mathcal{M}}$ sends $\frac{1}{6mx\alpha^*\beta_{\text{FCG}}}$ flow units to each other with no congestion.*

Proof: Assume otherwise. Then the value of the maximum concurrent flow in graph H_i for the set $\tilde{\mathcal{M}}$ of demands is less than $\frac{1}{6mx\alpha^*\beta_{\text{FCG}}}$.

We set up an instance of the non-uniform sparsest cut problem on graph H_j with the set $\tilde{\mathcal{M}}$ of demand pairs. Then there is a cut (A, B) in graph H_j, with $\frac{|E_{H_j}(A,B)|}{D_{H_j}(A,B)} < \frac{1}{6mx\alpha^*}$. Let (A', B') be the minimum

cut in graph G', where $A \subseteq A', B \subseteq B'$. Then $|E_{G'}(A',B')| = \mathrm{cut}_{G'}(A,B) \leq 2x\alpha^*|E_{H_j}(A,B)|$, while $D_{G'}(A',B') = D_{H_j}(A,B)$. Therefore, $\frac{|E_{G'}(A',B')|}{D_{G'}(A',B')} \leq 2x\alpha^* \frac{|E_{H_j}(A,B)|}{D_{H_j}(A,B)} < \frac{1}{3m}$. We next show that there is a concurrent flow, in graph G', of value $\frac{1}{3m}$ and no congestion, between the pairs of the fake terminals in $\tilde{\mathcal{M}}$. This contradicts the fact that the value of the sparsest cut in graph G' is less than $\frac{1}{3m}$.

In order to find the concurrent flow in graph G', it is enough to show that, for every pair $(v_Q, v_{Q'}) \in \tilde{\mathcal{M}}$ of fake terminals, we can send one flow unit from some vertex of Q to some vertex of Q' simultaneously with congestion at most $3m$ in graph G. Scaling this flow down by factor $3m$ will define the desired flow in graph G'.

Consider some pair $(v_Q, v_{Q'}) \in \tilde{\mathcal{M}}$ of fake terminals and let $(s,t) \in \mathcal{M}$ be the original demand pair that defined the pair $(v_Q, v_{Q'})$. Recall that there is a path $P \in \mathcal{P}$ connecting s to t, and paths $P_s, P_t \in \mathcal{R}''$, where P_s connects s to some vertex $u \in Q$, and P_t connects t to some vertex $u' \in Q'$. The concatenation of these three paths gives a path that connects u to u' in graph G. Since the paths in $\mathcal{P}$ cause congestion at most $2m$, while the paths in $\mathcal{R}''$ cause congestion at most 24, the total congestion caused by these flow-paths is at most $2m + 24 < 3m$. We conclude that there is a concurrent flow in graph G' of value $\frac{1}{3m}$ between the pairs of terminals in $\tilde{\mathcal{M}}$, contradicting our former conclusion that the value of the sparsest cut in G' is less than $\frac{1}{3m}$. □

In the rest of the algorithm, we apply the algorithm of Rao-Zhou to each of the graphs $H_1, \ldots, H_x$ in turn, together with some subset $\tilde{\mathcal{M}}^j \subseteq \tilde{\mathcal{M}}$ of the fake demand pairs. The output of the iteration is a collection $\tilde{\mathcal{P}}^j$ of edge-disjoint paths in graph H_j connecting some demand pairs in $\tilde{\mathcal{M}}^j$. We say that a pair $(\tilde{S}_i, \tilde{T}_i)$ is satisfied in iteration j, iff $\tilde{\mathcal{P}}^j$ contains at least $\frac{D}{48mx\alpha^*\beta_{\mathrm{FCG}}\alpha_{\mathrm{RZ}}}$ paths connecting demand pairs in $\tilde{\mathcal{M}}_i$.

We now fix some $1 \leq j \leq x$ and describe the execution of iteration j. Let $I \subseteq \{1, \ldots, k\}$ be the set of indices i, such that $(\tilde{S}_i, \tilde{T}_i)$ was not satisfied in iterations $1, \ldots, j-1$. Let $\tilde{\mathcal{M}}' = \bigcup_{i \in I} \tilde{\mathcal{M}}_i$. From Lemma 17, there is a fractional solution F in graph H_j, where every demand pair in $\tilde{\mathcal{M}}'$ sends $\frac{1}{6mx\alpha^*\beta_{\mathrm{FCG}}}$ flow units to each other with no congestion. We transform instance $(H_j, \tilde{\mathcal{M}}')$ of group-ICF into a canonical instance, obtaining, for each $i \in I$, a collection $\tilde{\mathcal{P}}'_i$ of $\lfloor \frac{D}{6x\alpha^*\beta_{\mathrm{FCG}}} \rfloor$ paths, such that the set $\bigcup_{i \in I} \tilde{\mathcal{P}}'_i$ of paths causes congestion at most $2m$ in graph H_j. Let $\tilde{\mathcal{M}}^j$ be the collection of pairs of endpoints of paths in $\bigcup_{i \in I} \tilde{\mathcal{P}}'_i$. We apply Theorem 24 to the EDP instance defined by the graph H_j and the set $\tilde{\mathcal{M}}^j$ of the demand pairs. Let $\tilde{\mathcal{P}}^j$ be the output of the algorithm. Then w.h.p., $|\tilde{\mathcal{P}}^j| \geq \frac{|\tilde{\mathcal{M}}^j|}{2m\alpha_{\mathrm{RZ}}}$, and the paths in $\tilde{\mathcal{P}}^j$ are edge-disjoint.

It is easy to verify that at least $\frac{1}{8m\alpha_{\mathrm{RZ}}}$-fraction of pairs $(\tilde{S}_i, \tilde{T}_i)$ for $i \in I$ become satisfied in iteration j. This is since $|\tilde{\mathcal{P}}^j| \geq \frac{|\tilde{\mathcal{M}}^j|}{2m\alpha_{\mathrm{RZ}}} \geq \frac{|I| \cdot D}{24m\alpha_{\mathrm{RZ}}x\alpha^*\beta_{\mathrm{FCG}}}$, each unsatisfied pair contributes at most $\frac{D}{48m\alpha_{\mathrm{RZ}}x\alpha^*\beta_{\mathrm{FCG}}}$ paths to $\tilde{\mathcal{P}}^j$, and each satisfied pair contributes at most $\frac{D}{6x\alpha^*\beta_{\mathrm{FCG}}}$ paths. Therefore, after $x = 16m\alpha_{\mathrm{RZ}} \cdot \log n$ iterations, all demand pairs are satisfied.

Let $\tilde{\mathcal{P}} = \bigcup_{j=1}^{x} \tilde{\mathcal{P}}^j$ denote the final collection of paths. For each demand pair $(\tilde{S}_i, \tilde{T}_i)$, set $\tilde{\mathcal{P}}$ contains at least $\frac{\tilde{D}_i}{48mx\alpha^*\beta_{\mathrm{FCG}} \cdot \alpha_{\mathrm{RZ}}} = \Omega\left(\frac{D_i}{\log^{42} n \operatorname{poly}\log\log n}\right)$ paths connecting the vertices of $\tilde{S}_i$ to the vertices of $\tilde{T}_i$, and the paths in $\tilde{\mathcal{P}}$ are edge-disjoint. Applying Lemma 16, we obtain a collection $\mathcal{P}^*$ of paths in graph G, that cause a constant congestion, and for each $1 \leq i \leq k$, at least $\Omega\left(\frac{D_i}{\log^{42} n \operatorname{poly}\log\log n}\right)$ paths connect the vertices of S_i to the vertices of T_i.

I Arbitrary Edge Capacities and Demands

In this section we extend our algorithms for basic-ICF and for group-ICF from Sections 4 and H to arbitrary demands and edge capacities. We only present here the generalization for basic-ICF, since the extension of the algorithm for group-ICF to general edge capacities and demands is almost identical.

Let $\alpha = \operatorname{poly}\log n$ denote the approximation factor of the algorithm from Section 4, and let γ denote the congestion. Recall that for each demand pair $(t,t') \in \mathcal{M}$, the algorithm finds $\lfloor \lambda_{\mathrm{OPT}} D/\alpha \rfloor$ paths connecting t to t' in G.

We now assume that we are given a set $\mathcal{D}$ of arbitrary demands, and the edge capacities are also arbitrary. We assume w.l.o.g. that $\lambda_{\mathrm{OPT}} \leq 1$. Let $D_{\max}$ and $D_{\min}$ be the maximum and the minimum demands in $\mathcal{D}$, respectively.

We first consider the case where $D_{\max}/D_{\min} \leq n^3$, and show a factor 4α-approximation with congestion at most γ for it. By scaling all demands and edge capacities by the same factor ρ, we can assume w.l.o.g. that the value of the minimum demand $D_{\min} = 2n^3$, and so the value of the maximum demand, $D_{\max} \leq 2n^6$. This change leaves the value λ_{OPT} unchanged, so $\lambda_{\text{OPT}} \leq 1$ still holds.

Let $D^* = 4\alpha/\lambda_{\text{OPT}}$. Since we are only interested in finding a factor 4α-approximation, we can assume w.l.o.g., that for each pair of terminals, either $D(t,t') = 0$, or $D(t,t') \geq D^*$. In particular, $D_{\min} \geq D^*$.

We now slightly modify the graph G, and define a new set $\mathcal{D}'$ of demands, such that in the new instance all capacities are unit, and the demands are uniform. Let $D = D_{\min}$. We start with $\mathcal{M} = \emptyset$. Consider some demand pair (s,t) with $D(s,t) > 0$, and let $N(s,t) = \lfloor D(s,t)/D \rfloor$. We create $N(s,t)$ copies of the source s that connect to s with a capacity-1 edge each, and $N(s,t)$ copies of the sink t that connect to t with capacity-1 edges. We also add $N(s,t)$ disjoint pairs of vertices, each containing one copy of s and one copy of t, to set $\mathcal{M}$. Let $\mathcal{M}$ be the final set of terminal pairs, obtained after we process all pairs with non-zero demands, and let $\mathcal{D}'$ be the corresponding set of demands, where for each pair $(s',t') \in \mathcal{M}$, we set its demand $D'(s',t')$ to be D. Notice that so far for each pair (s,t) of vertices with $D(s,t) > 0$, the total demand in $\mathcal{D}'$ between the copies of s and the copies of t is at least $D(s,t)/2$ and at most $D(s,t)$. Therefore, an α'-approximate solution to the resulting instance will give a $2\alpha'$-approximation to the original instance. Our final step is to take care of the non-uniform edge capacities. We remove all edges whose capacities are less than 1. Notice that the total amount of flow going through such edges in the optimal fractional solution is bounded by $n^3 \leq D/2$, so this change reduces the value of the optimal fractional solution by at most factor 2. Since $\lambda_{\text{OPT}} \leq 1$, the total flow through any edge in the optimal fractional solution cannot exceed $n^2 \cdot D_{\max}$, so if the capacity of any edge is above $n^2 \cdot D_{\max}$, we can set it to $n^2 \cdot D_{\max}$ without changing the value of the optimal fractional solution. Finally, for each edge e, we replace e with $\lceil c(e) \rceil$ parallel edges with unit capacities. The resulting instance of ICF has unit edge capacities and uniform demands. The value of the optimal fractional solution is at least $\lambda_{\text{OPT}}/2$, where λ_{OPT} is the value of the optimal fractional solution in the original instance. We can now use the algorithm from Section 4 to find an α-approximate integral solution with congestion at most γ for this new instance. This solution immediately gives a factor 4α-approximation with congestion at most 2γ for the original instance.

Assume now that we are given an instance $(G,\mathcal{D})$ of basic-ICF with arbitrary demands and capacities. Assume w.l.o.g. that all demands are at least 1. We group the demands geometrically into groups $\mathcal{D}_1, \mathcal{D}_2, \ldots$, where group $\mathcal{D}_i$ contains all demands $D(s,t)$ where $n^{3(i-1)} \leq D(s,t) < n^{3i}$. Notice that the number of non-empty groups $\mathcal{D}_i$ is bounded by n^2. For each non-empty group $\mathcal{D}_i$, we create a new instance of basic-ICF, as follows. We build a graph G_i whose set of vertices is $V(G)$, and the set of edges consists of all edges of G whose capacities are at least $n^{3(i-2)}$. If the capacity of an edge is more than n^{3i+2}, then we set its capacity to n^{3i+2}. The capacities of all other edges remain unchanged. We then use the algorithm for the special case where $D_{\max}/D_{\min} \leq n^3$ for each one of the resulting instances $(G_i,\mathcal{D}_i)$, and output the union of their solutions. Since the value of the optimal fractional solution in each such instance is at least $\lambda_{\text{OPT}}/2$, it is immediate to verify that we obtain a factor 8α-approximation. In order to bound the edge congestion, observe that for each edge $e \in E(G)$, the total capacity of copies of edge e in all instances $(G_i,\mathcal{D}_i)$ to which e belongs is bounded by $4c(e)$. Therefore, the overall edge congestion is bounded by 8γ.