HALLUCINATION HELPS: ENERGY EFFICIENT VIRTUAL CIRCUIT ROUTING

ANTONIOS ANTONIADIS*, SUNGJIN IM†, RAVISHANKAR KRISHNASWAMY‡, BENJAMIN MOSELEY§, VISWANATH NAGARAJAN¶, KIRK PRUHS‖, AND CLIFF STEIN#

Abstract. We consider virtual circuit routing protocols with an objective of minimizing energy in a network of components that are speed scalable, and that may be shutdown when idle. We assume the standard model for component power: the power consumed by a component with load (speed) s is $\sigma + s^\alpha$ where σ is the static power and the exponent $\alpha > 1$. We obtain a very simple $O(\log^\alpha k)$-approximation algorithm for multicommodity routing where k is the number of demand pairs. This improves upon previous results by several logarithmic factors. The key step in our algorithm is a random sampling technique that we call *hallucination*, which is reminiscent of the Sample-Augment framework for Buy-at-Bulk problems, and sampling in cut-sparsification algorithms.

We also consider the *online* setting of the problem, where demand pairs arrive over time. We show that our offline algorithm naturally extends to the online setting, and obtain a randomized competitive ratio of $\tilde{O}(\log^{3\alpha+1} k)$, which is the first non-trivial bound. The analysis of this algorithm involves the study of *priority multicommodity flows*, where edges and demand-pairs have priorities and each demand-pair must route its flow only on edges of lower priority. We establish a poly-logarithmic *flow-cut* gap for these priority flows, which we believe is of independent interest.

Finally, we show how our technique can be used to achieve a randomized $(O(\log m), O(\log^2 m))$ bi-criteria competitive algorithm for the *uniform capacitated network design* problem, where m is the number of edges. Here, every edge has a cost c_e and uniform capacity q, and the goal is to choose the minimum cost subgraph that can support the given multicommodity demand. This is the first online algorithm for this problem. In fact, our approach also improves prior results in the offline setting by several logarithmic factors.

Key words. network design, energy efficiency, approximation algorithms, online algorithms

AMS subject classifications. 68Q25

1. Introduction.

According to the US Department of Energy [1], data networks consume more than 50 billion kWH of energy per year, and a 40% reduction in wide-area network energy is plausibly achievable using network components that dynamically adjust their speed to be proportional to demand. Virtual circuit routing, in which each connection is assigned a reserved route in the network with a guaranteed bandwidth, is used by several network protocols to achieve reliable communication [32]. In this paper we consider virtual circuit routing protocols, with an objective of minimizing energy, in a network with *speed-scalable* edges.

The Energy-Efficient Routing Problem (EERP). The input consists of an undirected network $G = (V, E)$, scalar non-negative multipliers on edges c_e, and a common energy cost function $f(\cdot)$ defined as

* Saarland University and Max-Planck-Institute for Informatics. Supported in part by DFG grant AN 1262/1-1.

†University of California, Merced. S. Im was supported in part by NSF grants CCF-1409130, CCF-1617653, and CCF-1844939.

‡ Microsoft Research.

§Carnegie Mellon University. Research supported in part by NSF grants CCF-1733873, CCF-1845146, and CCF 1824303 and a Google Faculty Award

¶University of Michigan. Supported in part by NSF grant CCF-1750127.

‖ University of Pittsburgh. Supported in part by NSF grants CCF-1421508, CCF-1535755, CCF-1907673 and an IBM Faculty Award.

Columbia University. Research supported in part by NSF grants CCF-1421161 and CCF-1714818.

$$f(x) = \begin{cases} 0 & \text{if } x = 0 \\ \sigma + x^\alpha & \text{if } x > 0 \ . \end{cases} \quad . \tag{1.1}$$

We are also given a collection of k requests of the form (s_i, t_i, d_i), where for each $i \in [k] = \{1, 2, \cdots k\}$, $s_i \in V$ and $t_i \in V$ need to be connected by a flow path with dedicated capacity/bandwidth of $d_i \geq 1$ (called a virtual circuit). The objective is to find the flow paths P_i for each request i to minimize the overall energy cost of the routing defined as

$$\sum_{e \in E} c_e \cdot f\left(\sum_{i : e \in P_i} d_i\right).$$

We call this problem the *energy efficient routing problem* (EERP), and remark that this is precisely the problem formulation studied in [4, 3, 11]. In the offline setting, all requests are known in advance before selecting paths. In the more realistic (and harder) online setting, requests arrive over time and the algorithm needs to select a virtual circuit P_i for each request immediately upon arrival. We consider both the offline and online settings in this paper.

Why this energy function $f(\cdot)$? Even though the model is by now standard, (see, e.g., [3]), we provide a brief motivation. Speed-scalable network components (edges, in our case) are associated with a power-rate curve $f(x)$. This function measures the power consumption as a function of its speed x. The speed of a component is assumed to be proportional to the traffic load passing through the component, which in our case is the total bandwidth reserved on this edge.

In equation (1.1) above, the parameter σ is the *static power*. That is, the power used when the component is turned on but idle. The static power can only be saved by turning the component off, which only happens when its load/speed is 0. The term x^α is the *dynamic power* of the component as it varies with the speed, or equivalently load, of the component. Here $\alpha > 1$ is a parameter specifying the energy inefficiency of the components, as speeding up by a factor of s increases the energy used per unit computation/communication by a factor of $s^{\alpha-1}$. The value of α is in the range $[1.1, 3]$ for essentially all technologies [13, 41]. As in prior work [3, 11], we will assume that all components in the network are homogenous: so the parameters α and σ are uniform across all network components.

Relation to Buy-at-Bulk Network Design. The energy efficient routing problem has some similarity to the classic buy-at-bulk network design problem [8, 39]. The difference is that buy-at-bulk involves a *concave* edge cost-function instead of f as defined in (1.1). Note that if the cost function is concave, then it is always better to aggregate flow as much as possible. Indeed, the idea of aggregating flows forms the basis of all known algorithms for the buy-at-bulk problem. There are offline and online algorithms for buy-at-bulk with poly-logarithmic ratios under both uniform and non-uniform edge-costs [8, 19, 15].

In contrast, the edge cost function (1.1) is not concave (it is not convex either). In particular, the static power term in function f is concave, whereas the dynamic power term is convex. This requires an algorithm to balance the two opposing goals described below.

- *Effect of the static power.* The static power is zero when the flow $x = 0$ and σ when $x > 0$. As this is concave, it is best to aggregate flow. Indeed,

an optimal solution here is to route all flow over a minimum cost *Steiner forest* [2, 23] that connects the corresponding request-pairs. There is also an $O(\log n)$-competitive online algorithm for Steiner forest [27].

- *Effect of the dynamic power.* The dynamic power is x^α which is convex. In this case, it is better to dis-aggregate flow as much as possible. The greedy algorithm that routes each request along a path of minimum increase in cost tends to balance the flows on different edges, and is known to achieve an $O(1)$-competitive ratio for any constant α [6, 24].

Previous work by Andrews et al. [3] showed that these competing forces can be somewhat balanced by giving a polynomial-time poly-logarithmic approximation algorithm for EERP. In this paper, we provide a simpler and better approximation algorithm that also extends to the online setting and other related problems.

1.1. Our Results. We present three main results in this paper, starting with the offline EERP problem.

THEOREM 1.1. *There is an efficient randomized $O(\log^\alpha k)$-approximation algorithm for the energy efficient routing problem.*

This algorithm improves over the previously known approximation algorithm from [3] in the following ways: (a) the approximation ratio is better by several $\log^\alpha k$ factors, (b) the algorithm is itself very simple to describe and implement, and (c) the analysis is also considerably simpler, with the only real "hammer" being the classic flow-cut gap for multicommodity flow. Moreover, our techniques extend naturally to the online setting, which is our second main result.

THEOREM 1.2. *There is an efficient randomized $\tilde{O}(\log^{3\alpha+1} k)$-competitive online algorithm for the energy efficient routing problem.*

This is the first non-trivial online algorithm for this basic energy minimization problem. Previous results in the online setting could only handle the single-commodity special case [11].

Finally, we consider the seemingly unrelated problem of *capacitated network design* (CapND). Here, we are given a graph $G = (V, E)$ with n vertices and m edges, where each edge $e \in E$ is associated with a cost $c_e \geq 0$ and uniform capacity of $q \geq 0$. There are k requests of the form (s_i, t_i, d_i) as in EERP. The goal is to choose a *minimum cost* subgraph $H \subseteq E$ that can support the multicommodity flow requirement of the requests (concurrently). In the online version, requests arrive online and the algorithm must buy edges irrevocably to support the evolving flow requirements.

THEOREM 1.3. *There is a randomized $(O(\log m), O(\log^2 m))$ bi-criteria competitive online algorithm for uniform capacitated network design, i.e., with high probability, the solution costs $O(\log m)$ times the optimum and violates edge capacities by factor $O(\log^2 m)$.*

Again, this is the first non-trivial online algorithm for CapND. In fact, this approach also improves significantly over the offline algorithm in [3] which had a (large) poly-logarithmic bi-criteria approximation ratio. In the conference version [5] of this paper, we described an offline algorithm with a slightly better $(O(\log m), O(\log m))$ bi-criteria approximation ratio. Here, we only focus on the (more general) online setting.

1.2. Our Techniques. As discussed above, any algorithm for EERP needs to balance the effects of aggregating and dis-aggregating flows. Intuitively, when the static power is larger than the dynamic power (i.e. $\sigma > x^\alpha$ where x is the flow on an edge) then we want to aggregate flows. On the other hand, when the dynamic power is

larger (i.e. $\sigma < x^\alpha$) then we want to dis-aggregate flows. This suggests the approach of (i) aggregating demands until the total demand exceeds the threshold $q := \sigma^{1/\alpha}$, and then (ii) routing these "composite demands" in a dis-aggregated manner. Indeed, this is the approach used in [3]. We use a different approach: instead of explicitly aggregating demands, we use *random sampling* of the requests to help identify edges with large flow.

For simplicity, consider the case when all demands are unit. In this case, our algorithm proceeds as follows. First, the algorithm selects an approximate minimum cost Steiner forest G_S to ensure minimal connectivity between all request-pairs. Second, each request-pair, with probability $\frac{O(\log k)}{q}$, *hallucinates* that it wants to route q units of flow instead of one.[1] When every demand is at least q, the edge costs (1.1) are always dominated by the dynamic power: so we can use existing algorithms [6, 24] for minimizing just the dynamic power. This yields another subset G_H of edges, i.e. all edges used in the routing of hallucinated demands. Finally, we route all the original (unit) demands on the subgraph $G_S \cup G_H$ so as to minimize just the dynamic power: here we use the algorithm from [6, 24] again. The key steps in the analysis are in showing that (i) the hallucinated demands can be routed at low cost (this is a simple randomized rounding argument) and (ii) the original demands can be routed at low cost on the chosen subgraph $G_S \cup G_H$ (this uses results on cut-sparsification and flow-cut gaps).

The hallucination technique is rather similar to the Sample-Augment framework [25] for solving Buy-at-Bulk type problems. This is perhaps surprising because in Buy-at-Bulk, the cost on edges is purely concave, whereas in our case the cost is convex after the jump at 0. The similarities stop there, as the analyses are very different for the two problems: our analysis more closely resembles those of cut-sparsification algorithms [29].

A striking benefit of this simple offline algorithm is that it directly extends to the online setting. Indeed, there are good online algorithms for both Steiner forest [27] and dynamic-power minimization [6, 24], which can be used directly. The online algorithm's analysis however is considerably more involved than the offline case, and we believe that the techniques introduced here are of independent interest. The major difference (compared to the offline setting) is that the subgraph $G_S \cup G_H$, which is used in actually routing the demands, is built incrementally over time. Therefore, we are faced with a non-standard multicommodity flow problem, that we call *priority multicommodity flow*. Here, each edge comes with a priority (indicative of the time when it was chosen by our online algorithm), and each request also comes with a priority (indicative of its arrival time), and a request can only use edges with priority lower than or equal to itself. So the key question we are interested in is: under what conditions is there a good concurrent priority multicommodity flow? To this end, we introduce the notions of *priority-cuts* and *prefix-sparsity*, and establish relationships between these quantities and priority multicommodity flows. In particular, we show that the values of priority multicommodity flow, priority-cut and prefix-sparsity are all within a poly-logarithmic factor of each other. In proving these results, we use a variant of region growing [33, 21] as well as new charging arguments.

Finally, we show that the hallucination approach also works for the CapND problem. Indeed, the relevant sampling parameter q for CapND is the uniform edge ca-

[1] We assume here that $q = \omega(\log k)$, so the hallucination probability is well-defined. The case $q = O(\log k)$ is much easier because the static power is always $O(\log^\alpha k)$ times the dynamic power, which means we can directly use existing algorithms [6, 24] for minimizing the dynamic power.

pacity (instead of $\sigma^{1/\alpha}$ in the EERP algorithms). The CapND instance consisting of hallucinated demands (each of q units) can be solved online using an existing algorithm for *min-cost circuit routing* [9]. Moreover, the analysis of this online algorithm is very similar to the offline EERP algorithm.

1.3. Related Work. The energy efficient routing problem was introduced in [4] and a poly-logarithmic approximation algorithm was obtained in [3]. It is also known (using the relationship to buy-at-bulk) that EERP is hard to approximate to within an $\Omega(\log^{1/4-\epsilon} n)$ factor [3]. The high-level algorithmic strategy in [3] is to aggregate the flow within suitably defined groups, such that each group contains a total crossing demand of roughly q. Then, they find a low-cost routing across groups. Roughly, the flow paths going across groups carry large loads, so the dynamic power dominates; whereas, flow paths within each group carry small loads, so the static power dominates. However, the algorithm design to achieve such a routing as well as its analysis in [3] are fairly complicated and rely on various tools: the well-linked decomposition of Chekuri-Khanna-Shepherd [20], the construction of expanders via matchings of Khandekar-Rao-Vazirani [30], and edge-disjoint routings in well-connected graphs due to Rao-Zhou [37]. Moreover, the exponent of $\log^{\alpha} k$ in the poly-log approximation ratio is sufficiently large that it was not explicitly calculated in [3].

Bansal et al. [11] considered EERP in the case of a common source vertex s for all request-pairs, that is all $s_i = s$. Applications for a common source vertex include data collection by base stations in a sensor network, and supporting a multicast communication using unicast routing. In this single-commodity setting, [11] gave an $O(1)$-approximation algorithm. The algorithm and analysis are considerably easier than [3] because, after aggregation into groups, all the flow is going to the same place. [11] also gave an $O(\log^{2\alpha+1} n)$-competitive randomized online algorithm, by giving a procedure for forming groups in an online fashion.

The uniform CapND problem was also studied in [3], where a poly-logarithmic bicriteria approximation algorithm was obtained. In fact, [3] used this result as a sub-routine for their overall algorithm for EERP. In this paper, we show that we can use the hallucination approach to get a simpler and improved algorithm for CapND which also extends to the online setting.

A related problem to the CapND problem we consider is *capacitated survivable network design* [14, 17, 26], where the requirement is to compute a minimum cost subgraph H which satisfies the flow requirement *individually for each demand* rather than concurrently. These results are incomparable to those for CapND.

Finally, *priority* versions of a number of classic problems have been studied in approximation algorithms, e.g. priority Steiner tree [18] and priority covering integer programs [16]. We note however that our focus in the priority multicommodity flow problem is structural (bounding its flow-cut gap) rather than algorithmic.

2. Preliminaries. We begin by recalling the problem statement. The energy efficient routing problem (EERP) is defined on an undirected multi-graph $G = (V, E)$ with $|V| = n$ vertices and m edges. Each edge e is associated with a scaling factor $c_e \geq 0$. There are k request-pairs, where the i^{th} pair specifies a source s_i, a destination t_i and an integer demand $d_i \geq 1$. We need to route each request-pair unsplittably so as to minimize the objective $\sum_{e \in E} c_e \cdot f(\ell_e)$ where ℓ_e denotes the flow (i.e. total demand routed) on edge e and function f is as defined in (1.1).

In the analysis we will also be concerned with splittable (fractional) routings, where the demand of each request may be satisfied using multiple paths. Unless

specified otherwise, any routing in solutions we seek is assumed to be unsplittable.

The power incurred on any edge $e \in E$ is naturally split into two parts: (i) the *static power* which is $c_e\sigma$ if it routes any flow or 0 otherwise, and (ii) the *dynamic power* which is $c_e\ell_e^\alpha$ where ℓ_e denotes the total demand routed on edge $e \in E$. The static (resp. dynamic) power of a routing is the sum over all edges of the static (resp. dynamic) power.

As mentioned earlier, a useful parameter throughout the paper is $q := \sigma^{1/\alpha}$, which is the amount of flow on an edge for which the static and dynamic power are equal. We use Opt to denote the total power of a fixed optimal solution. We assume that $\alpha \geq 1$ is a constant, so any function of α is just $O(1)$.

Moreover, we assume that $q = \omega(\log k)$, as otherwise it is easy to obtain an $O(\log^\alpha k)$-competitive online algorithm for EERP. Indeed, when $q = O(\log k)$, we have $\sigma = O(\log^\alpha k)$ and the static power of any routing is at most $O(\log^\alpha k)$ times the dynamic power (recall that all demands are integer). In this case, we could simply optimize for the dynamic power and obtain an approximation factor that is worse by a factor of $O(\log^\alpha k)$. As there are $O(1)$-competitive online algorithms for dynamic power minimization (see below), we would then immediately get an $O(\log^\alpha k)$-competitive online algorithm for EERP for the case $q = O(\log k)$.

Dynamic Power Minimization (DynPM). A crucial sub-routine in our algorithms is the so-called *waterfilling algorithm* [24] for the problem of minimizing just the dynamic power of the routing. The input to DynPM is the same as for EERP. We need to route each request-pair unsplittably so as to minimize the *dynamic power* objective $\sum_{e\in E} c_e(\ell_e)^\alpha$ where ℓ_e denotes the total demand routed on edge e. The waterfilling algorithm is a natural online greedy algorithm for this problem, which routes the demand of each request along the path that results in the smallest increase in the objective. An important feature of this algorithm is that we can also implicitly specify a subset $\mathcal{P}_i$ of allowed $s_i - t_i$ paths for each request (s_i, t_i, d_i). This will be useful in the online EERP algorithm, where the underlying graph is built incrementally, and each request can only use the edges present at the time of its arrival.

Online Waterfilling Algorithm for DynPM.
When request i arrives:
1. Let ℓ_e denote the current load on each edge e in graph G.
2. Choose $s_i - t_i$ path $P_i \in \mathcal{P}_i$ in G to minimize $\sum_{e\in P_i} c_e\left((\ell_e + d_i)^\alpha - \ell_e^\alpha\right)$.

THEOREM 2.1. *Given any graph $G = (V, E)$, and requests (s_i, t_i, d_i) with a set of allowed paths $\mathcal{P}_i$, the waterfilling algorithm is $O(\alpha^\alpha)$-competitive for the objective of minimizing the total dynamic power.*

In all our applications of Theorem 2.1, each set $\mathcal{P}_i$ (allowed $s_i - t_i$ paths) consists of all $s_i - t_i$ paths in some given *subgraph* of G. Note that in this case, the min-cost path $P_i \in \mathcal{P}_i$ in Step 2 above can be computed in polynomial time by running any shortest path algorithm on the given subgraph.

For the case where $\mathcal{P}_i$ is the set of *all* s_i-t_i paths in G, [24] proved Theorem 2.1 using a dual-fitting framework. Previously, [10] used a potential function based framework to prove a similar result in the special case of the load balancing problem. It is not hard to adapt these existing analyses to our setting where each request also specifies a set of allowed paths. For completeness, we give a proof of Theorem 2.1 in Appendix A using the approach from [10].

3. Offline Algorithm for EERP. In this section we give a polynomial time $O(\log^\alpha k)$-approximation algorithm for EERP, thereby proving Theorem 1.1. As men-

tioned in Section 2 we may assume that $q = \omega(\log k)$: this makes the algorithm and analysis simpler. Below we use $\lambda = O(\log k)$ where the constant term will be set later such that $\frac{\lambda}{q} \leq 1$. We also assume that each demand $d_i \leq q$: again this is because demands larger than q can be routed using the algorithm for minimizing just dynamic power, and combining the two solutions will only incur an extra multiplicative factor of 2^α in the routing cost.

Our overall algorithm comprises two stages. In the first stage (steps 1 and 2 below) we decide which edges to power on or "buy" and incur the static cost. In the second stage (step 3 below) we route all the requests using the bought edges, to minimize just the dynamic power of the routing.

Offline Algorithm Description.
1. Constructing the Steiner backbone G_S. Solve the Steiner forest instance on graph $G = (V, E)$ with edge-costs $\{c_e : e \in E\}$ and pairs $\{(s_i, t_i)\}_{i=1}^k$ using the 2-approximation algorithm from [2, 23]. Let G_S denote the resulting solution.
2. Constructing the Hallucination backbone G_H. Each request $i \in [k]$ independently "hallucinates" a demand of $q \cdot B_i$ units, where $B_i \sim \mathsf{Binomial}(d_i, \frac{\lambda}{q})$. Let $\mathcal{I}_{hal}$ denote the resulting (random) instance of DynPM on graph G and the hallucinated demands. Run the waterfilling algorithm (Theorem 2.1) by feeding the requests in $\mathcal{I}_{hal}$ in an arbitrary order, with $\mathcal{P}_i$ being the set of all s_i-t_i paths in G. Let $\mathcal{H}$ denote the resulting unsplittable routing and let G_H denote the subgraph consisting of all edges used in $\mathcal{H}$.
3. Routing on the backbone. Let $\mathcal{I}_{act}$ denote the DynPM instance on graph $G_F = G_S \cup G_H$ with all the original requests $\{(s_i, t_i, d_i) : i \in [k]\}$. Feed the requests of $\mathcal{I}_{act}$ to the waterfilling algorithm (Theorem 2.1) in any order to obtain the final unsplittable routing $\mathcal{R}$.

Note that the hallucinated flow $\mathcal{H}$ is used solely to determine which edges to consider for the final routing in step 3.

3.1. Analysis. Let Opt denote the optimal cost of the EERP instance. The cost of the algorithm is bounded by the total static power of the backbone $G_F = G_S \cup G_H$ (from Steps 1 and 2) plus the total dynamic power of routing $\mathcal{R}$ (from Step 3).
Static power of G_S. The static power of any feasible EERP solution is clearly at least σ times the optimal Steiner forest cost. So, using the 2-approximation algorithm for Steiner forest [2, 23], it follows that the static power for the edges in G_S is at most twice that in the EERP optimum, i.e. at most $2 \cdot \mathsf{Opt}$.
Static power of G_H. The static power of the hallucination backbone G_H is at most the dynamic power of the hallucinated flow $\mathcal{H}$ since every hallucinated request-pair routes at least q units of flow unsplittably in $\mathcal{H}$. In Lemma 3.1 we show that the dynamic power (and hence static power) for the hallucinated flow is $O(\lambda^\alpha) \cdot \mathsf{Opt}$ using a simple probabilistic argument. A similar idea was used in [11] for the online single-commodity EERP.
Dynamic power. In order to bound the dynamic power of our routing in Step 3, by Theorem 2.1 it suffices to prove that there *exists* a routing of low dynamic power in the subgraph G_F. To this end, we assign each edge in the backbone G_F a *capacity* $\widehat{q}_e$ equal to the amount of hallucinated flow routed on it in $\mathcal{H}$, plus λq if it is in the Steiner backbone G_S. Importantly, using the dynamic power of $\mathcal{H}$, these capacities will satisfy $\sum_{e \in E} c_e \widehat{q}_e^\alpha = O(\lambda^\alpha) \cdot \mathsf{Opt}$. Then we show (in Lemma 3.3) that the non-uniform sparsest cut (w.r.t the demands of all requests in the original instance) under these capacities is $\Omega(\log k)$ with high probability: this uses a classic cut sparsification

result [29]. Next, using the $O(\log k)$ flow-cut gap for multicommodity flow [33, 34, 7], we obtain the existence of a *fractional* routing within the backbone G_F that respects capacities c. Finally, in Lemma 3.5 we show by randomly rounding this fractional flow (as in [24]) that there exists an integral routing in G_F with dynamic power $O(1)$ times the dynamic power of the fractional routing.

LEMMA 3.1. *The expected dynamic power of hallucinated flow $\mathcal{H}$ is $O(\lambda^\alpha) \cdot \mathsf{Opt}$.*

Proof. Consider any fixed optimal EERP solution that routes each request $i \in [k]$ along some path P_i^*. Let $\mathcal{O}$ denote the routing that sends the hallucinated demand $q \cdot B_i$ along path P_i^* for each $i \in [k]$. We will show that the expected dynamic power of $\mathcal{O}$ is $O(\lambda^\alpha) \cdot \mathsf{Opt}$. Combined with Theorem 2.1, it would follow that the expected dynamic power of the hallucinated flow $\mathcal{H}$ is at most $O(\lambda^\alpha) \cdot \mathsf{Opt}$.

We bound the expected dynamic power in $\mathcal{O}$ separately for each edge $e \in E$. Fix an edge e and let $K \subseteq [k]$ denote the requests whose optimal paths P_i^* use e. Let $N = \sum_{i \in K} d_i$ be the load on e in the optimal EERP solution; so the cost of edge e in this solution is $c_e(\sigma + N^\alpha)$. The load on edge e in $\mathcal{O}$ is $M = \sum_{i \in K} q \cdot B_i$. Note that we can write $B_i = \sum_{j=1}^{d_i} X_{ij}$ where each $X_{ij} \sim \mathsf{Bernoulli}(\frac{\lambda}{q})$ independently. So random variable M is the sum of N independent random variables that each take value q with probability λ/q (and zero otherwise). Using Corollary B.3 with $p = \lambda/q$ and $D = q$, we obtain:

$$\mathbb{E}[M^\alpha] \leq O(1) \left(\frac{\lambda}{q} \cdot N \cdot q^\alpha + (\frac{\lambda}{q} \cdot N \cdot q)^\alpha \right) \leq O(\lambda^q) \cdot \left(N q^{\alpha-1} + N^\alpha \right).$$

Note that $Nq^{\alpha-1} \leq \max\{N, q\}^\alpha \leq N^\alpha + q^\alpha = N^\alpha + \sigma$. Combined with the above inequality, we obtain $\mathbb{E}[M^\alpha] = O(\lambda^q) \cdot (2N^\alpha + \sigma)$, and so the cost of the edge e in $\mathcal{H}$ is $O(\lambda^q)$ times the cost of edge e in the optimal EERP solution.

Now summing over all edges $e \in E$ and using linearity of expectations, we conclude that the expected dynamic power in $\mathcal{O}$ is at most $O(\lambda^\alpha) \cdot \mathsf{Opt}$. ∎

LEMMA 3.2. *The expected static power of the backbone, G_F is $\sigma \cdot \mathbb{E}[\sum_{e \in G_F} c_e] = O(\lambda^\alpha) \cdot \mathsf{Opt}$.*

Proof. Observe that the static power of the backbone,

$$\sigma \sum_{e \in G_F} c_e \leq \sigma \sum_{e \in G_S} c_e + \sigma \sum_{e \in G_H} c_e \leq 2\mathsf{Opt} + \sigma \sum_{e \in G_H} c_e$$

as G_S is a 2-approximate Steiner forest. So the expected static power $\sigma \cdot \mathbb{E}[\sum_{e \in G_F} c_e] \leq 2\mathsf{Opt} + \sigma \mathbb{E}[\sum_{e \in G_H} c_e]$. It now suffices to bound $\sigma \mathbb{E}[\sum_{e \in G_H} c_e]$ which is the expected static power of $\mathcal{H}$. Note that if an edge has positive load in $\mathcal{H}$ then it has load at least q. So the static power of $\mathcal{H}$ is at most its dynamic power. Combined with Lemma 3.1 we obtain $\sigma \mathbb{E}[\sum_{e \in G_H} c_e] = O(\lambda^\alpha) \cdot \mathsf{Opt}$, which proves the lemma. ∎

It remains to bound the dynamic power of routing all demands in the backbone G_F. To this end, we assign a virtual capacity of $q\lambda$ on all edges in G_S, and for edges in G_H, we set the virtual capacity to be equal to the load e carries in $\mathcal{H}$ (which is at least q). Note that if we show that there exists a flow which respects these capacities routing all demands, then the dynamic power of such a routing would be at most $O(\lambda^\alpha)$ times the static power of G_S plus the dynamic power of G_H, both of which are bounded in the previous lemmas. Indeed, this is what we show by establishing good bounds on the *sparsity* of the graph with these virtual capacities w.r.t. the demands

of the request pairs. More formally, let $\{H_e : e \in E\}$ denote the edge loads induced by the routing $\mathcal{H}$. Then we define the following edge capacities:

$$\widehat{q}_e = \begin{cases} H_e & \text{if } e \in G_F \setminus G_S \\ H_e + \lambda q & \text{if } e \in G_S \end{cases}, \qquad \forall e \in E.$$

Note that only edges in $G_F = G_S \cup G_H$ have nonzero capacity.

Moreover, $\widehat{q}_e^{\alpha} \leq 2^{\alpha} \left(H_e^{\alpha} + \lambda^{\alpha}\sigma \cdot \mathbf{1}_{e \in G_S}\right)$, so we have:

$$\sum_{e \in E} c_e \mathbb{E}[\widehat{q}_e^{\alpha}] \leq 2^{\alpha} \left(\sum_{e \in E} c_e \mathbb{E}[H_e^{\alpha}] + \sigma\lambda^{\alpha} \cdot \sum_{e \in G_S} c_e \right) = O(\lambda^{\alpha}) \cdot \mathsf{Opt}, \tag{3.1}$$

where the last inequality uses (i) the static power of G_S is $\sigma \cdot \sum_{e \in G_S} \leq 2 \cdot \mathsf{Opt}$ and (ii) the expected dynamic power of $\mathcal{H}$ is $\sum_{e \in E} c_e \mathbb{E}[H_e^{\alpha}] = O(\lambda^{\alpha}) \cdot \mathsf{Opt}$ by Lemma 3.1.

Now we turn our attention to establishing lower bounds on the graph sparsity. For an undirected graph $G = (V, E)$ and subset $S \subseteq V$, we use the standard notation $\delta_G(S) := \{(u, v) \in E : u \in S, v \notin S\}$ for the *cut* corresponding to S. We shall sometimes refer to the vertices of these request-pairs as *terminals* to distinguish them from Steiner vertices in G that do not participate in any request-pair. The *sparsity* of a graph G with edge capacities $\widehat{q} : E \to \mathbb{R}_+$ w.r.t. the demands of all the request-pairs is the minimum (over all $S \subseteq V$) of the ratio of the capacity crossing cut S to the demand crossing it, i.e.

$$sparsity(G) := \min_{S \subseteq V} \frac{\sum_{e \in \delta(S)} \widehat{q}_e}{\sum_{i \in [k] : |S \cap \{s_i, t_i\}| = 1} d_i}.$$

It is well known that if the sparsity is $\Omega(\log k)$ then there is a fractional routing for all the requests that respects the capacities [34, 7].

LEMMA 3.3. *With probability at least $1 - k^{-3\alpha}$, the sparsity of graph G_F with edge capacities $\{\widehat{q}_e : e \in E\}$ and requests $\{(s_i, t_i, d_i) : i \in [k]\}$ is at least $\lambda/3$.*

Proof. For the proof we consider a *virtual* graph $\mathcal{B}$ on vertices V with the following edges and capacities:

- *Steiner* edges: each edge $e \in G_S$ has capacity $\bar{q}_e = \lambda q$.
- *Hallucinated* edges: for each $i \in [k]$ edge (s_i, t_i) has capacity $\bar{q}_{(s_i,t_i)} = q \cdot B_i$.

Each hallucinated edge (s_i, t_i) in $\mathcal{B}$ corresponds to an $s_i - t_i$ path carrying $q \cdot B_i$ flow in $\mathcal{H}$. Hence, for any $T \subseteq V$, the $\widehat{q}$-capacity of cut $\delta(T)$ is at least as much as its $\bar{q}$-capacity. Thus it suffices to show that the sparsity of $\mathcal{B}$ is at least λ.

We observe that the connected components in $\mathcal{B}$ are the same as those in the Steiner forest G_S: this is because every request pair $\{s_i, t_i\}_{i=1}^k$ is already connected in G_S. Moreover, in order to lower bound the sparsity of $\mathcal{B}$, it suffices to lower bound the sparsity of each component of G_S: this is because there are no requests across components of G_S. In particular, we will show that the sparsity of any component of G_S is at least λ with probability $1 - \frac{1}{k^{4\alpha}}$. Then, a union bound over all components in G_S (which are at most k) would prove the lemma.

Consider now any connected component of G_S. To reduce notation, we assume in the rest of the proof that there is a *single* component in G_S which connects all k pairs. (Otherwise, exactly the same argument works by restricting to the request-pairs in a particular connected component.) Let $G_S = (V_S, E_S)$ denote this connected component, which is a Steiner tree on all terminals. By shortcutting over degree two

Steiner vertices, we may assume that the number of vertices $|V_S|$ is at most $4k$ (i.e. $2k$ terminals and at most $2k$ Steiner vertices).

The main idea now is to apply a classic cut-sparsification result of [29]. To this end, consider a random multigraph $\mathcal{M}$ on vertices V_S by independently sampling the following edges:

- For each $e \in E_S$ there are λ parallel edges $e_1, \cdots e_\lambda$ between the end-points of e, each with probability $p(e_j) = 1$.
- For each request $i \in [k]$ there are d_i parallel edges $r_{i,1}, \cdots r_{i,d_i}$ between s_i and t_i, each with probability $p(r_{i,j}) = \lambda/q$.

Note that scaling all edges in $\mathcal{M}$ by factor q yields graph $\mathcal{B}$. Let $\widehat{\mathcal{M}}$ denote the weighted multigraph where each edge e has weight equal to its probability p_e (defined above). As E_S corresponds to a tree on all vertices V_S, the minimum cut in $\widehat{\mathcal{M}}$ is at least λ. By choosing $\lambda \geq 9(4\alpha + 2) \ln |V_S|$ which is $O(\log k)$, and applying Theorem 2.1 from [29], we obtain:

> With probability at least $1 - k^{-4\alpha}$, *every* cut in $\mathcal{M}$ has capacity at least $\frac{1}{3}$ times its capacity in $\widehat{\mathcal{M}}$.

Now, observe that the capacity of any cut $T \subseteq V_S$ in $\widehat{\mathcal{M}}$ is at least

$$\frac{\lambda}{q} \sum_{i \in [k]: |\{s_i, t_i\} \cap T| = 1} d_i \quad = \quad \frac{\lambda}{q} \cdot d(\delta T),$$

where $d(\delta T)$ denotes the total demand crossing cut T. Combined with the above cut-sparsification result and the fact that $\mathcal{B} \sim q \times \mathcal{M}$, it follows that the capacity of any cut $T \subseteq V_S$ in $\mathcal{B}$ is at least $\frac{\lambda}{3} \cdot d(\delta T)$ with probability at least $1 - k^{-4\alpha}$. Thus the sparsity of $\mathcal{B}$ is at least $\frac{\lambda}{3}$ with probability at least $1 - k^{-4\alpha}$. ∎

COROLLARY 3.4. *With probability at least $1 - k^{-3\alpha}$, there exists a fractional routing of all request-pairs in backbone G_F that respects edge capacities $\widehat{q}$.*

Proof. By Lemma 3.3 we know that the sparsity of G_F is at least $\lambda/3$. We also know that the flow-cut gap for concurrent multicommodity flow is $\rho = O(\log k)$. Hence, choosing $\lambda \geq 3 \cdot \rho$ it follows that there exists a feasible fractional routing. ∎

LEMMA 3.5. *The expected dynamic power of the routing $\mathcal{R}$ is at most*

$$O(1) \times \left(\mathbb{E}[\textit{Static power of } G_F] \; + \; \sum_e c_e \mathbb{E}[\widehat{q}_e^{\,\alpha}] \right) \; \leq \; O(\lambda^\alpha) \cdot \mathsf{Opt}.$$

Proof. We will bound the expected optimal dynamic power of instance $\mathcal{I}_{act}$ by $O(\lambda^\alpha) \cdot \mathsf{Opt}$. The lemma would then follow from Theorem 2.1.

We first consider the case where there exists a fractional routing $\mathcal{F}$ in G_F that respects capacities $\widehat{q}$ (which happens with probability at least $1 - k^{-3\alpha}$). In this case, the dynamic power of the fractional routing $\mathcal{F}$ is at most $\sum_{e \in E} c_e \widehat{q}_e^{\,\alpha}$. We now construct an unsplittable routing $\mathcal{U}$ from $\mathcal{F}$ by simple randomized rounding. For each request i, we take a path decomposition $\{P_j^i, \mu_j^i\}$ of the $s_i - t_i$ flow in the fractional routing; here μ_j^i is the fraction of i's demand routed along path P_j^i. Then we route d_i units along path P_i chosen independently from the distribution $\{P_j^i, \mu_j^i\}$, for each $i \in [k]$. For each edge e, let f_e (resp. U_e) denote the load on e in routing $\mathcal{F}$ (resp. $\mathcal{U}$). Note that $U_e = \sum_{i=1}^k d_i \cdot I_{e,i}$ where $I_{e,i} = \mathbf{1}_{e \in P_i}$. Also $f_e = \mathbb{E}[U_e]$. For a fixed edge e,

as the random variables $I_{e,i}$ are independent, we can use Theorem B.2 to obtain:

$$\mathbb{E}[U_e^\alpha] \le O(1) \cdot \max\{f_e^\alpha\,,\, \sum_{i=1}^{k} d_i^\alpha \cdot \mathbb{E}[I_{e,i}]\} \le O(1) \cdot \max\{f_e^\alpha\,,\, q^{\alpha-1} \cdot f_e\}.$$

The last inequality uses the fact that $\max_{i\in[k]} d_i \le q$. Moreover,

$$q^{\alpha-1} \cdot f_e \le \max\{q, f_e\}^\alpha \le q^\alpha + f_e^\alpha = \sigma + f_e^\alpha,$$

which implies $\mathbb{E}[U_e^\alpha] \le O(1) \cdot (\sigma + f_e^\alpha)$. So the expected dynamic power of $\mathcal{U}$ is

$$\begin{aligned}\sum_{e\in G_F} c_e \mathbb{E}[U_e^\alpha] &\le O(1) \sum_{e\in G_F} c_e(\sigma + f_e^\alpha)\\ &= O(1)\cdot\sigma \sum_{e\in G_F} c_e + O(1)\cdot \sum_{e\in G_F} c_e f_e^\alpha\\ &\le O(1)\cdot\sigma \sum_{e\in G_F} c_e + O(1)\cdot \sum_{e\in E} c_e \widehat{q}_e^\alpha\end{aligned}$$

The first inequality uses the fact that only edges of G_F are used in the routing $\mathcal{U}$ and the last inequality uses our assumption that $\mathcal{F}$ respects capacities $\widehat{q}$. Taking expectation over the random choices in Step 2, we obtain

$$\mathbb{E}[\sum_{e\in G_F} c_e U_e^\alpha] \quad \le \quad O(1)\cdot\sigma\mathbb{E}[\sum_{e\in G_F} c_e] + O(1)\cdot\sum_{e\in E} c_e \mathbb{E}[\widehat{q}_e^\alpha].$$

This proves the first part of the lemma (note that the first term in the right-hand-side above is the static power of G_F). Finally, using Lemma 3.2 and (3.1) the right-hand-side above is at most $O(\lambda^\alpha)\cdot \mathsf{Opt}$.

Now we consider the case that there is no capacity-respecting fractional routing in G_F. By Corollary 3.4 this occurs with probability at most $k^{-3\alpha}$. In this case, consider the routing that sends d_i demand along the unique $s_i - t_i$ path in G_S for each $i \in [k]$. Since each edge is used at most k times and $\max d_i \le q$, the dynamic power is at most k^α times the static power used by G_S. So the dynamic power of this routing is at most $2k^\alpha \cdot \mathsf{Opt}$. As this case only occurs with probability at most $k^{-3\alpha}$ (Corollary 3.4), the *expected* dynamic power of this routing is $k^{-3\alpha} \cdot 2k^\alpha \cdot \mathsf{Opt} = O(1)\cdot\mathsf{Opt}$. ∎

4. Priority Multicommodity Flows and Priority Cuts. We now take a detour, and describe a generalization of multicommodity flows which will help us in the analysis of our online algorithm for EERP. We stress that the online algorithm itself remains very simple. The following abstractions will be used exclusively in the analysis. We also believe that these *priority extensions* of the standard multicommodity flows would be of independent interest.

In a *priority multicommodity flow*, we are given an increasing sequence of multigraphs $G(1) \subseteq G(2) \subseteq \cdots \subseteq G(k)$ with respective requests $\{(s_i, t_i, d_i)\}_{i=1}^k$. Note that the vertex-set V remains the same for all the multigraphs. Also, while in general the edges could have arbitrary non-negative capacities, we assume for simplicity that all edges have unit capacity. Indeed, this is without loss of generality as we can replace an edge e with capacity $\widehat{q}_e$ with $\widehat{q}_e$ parallel edges of unit capacity. Hence, for the rest of this section, we assume that all edges have unit capacity.

DEFINITION 4.1 (Priority Multicommodity Flow). *Consider any sequence of multigraphs $G(1) \subseteq G(2) \subseteq \cdots \subseteq G(k)$ and requests $\{(s_i, t_i, d_i) : i \in [k]\}$. A priority*

multicommodity flow of value γ consists of a fractional routing of $\gamma \cdot d_i$ units of flow between s_i and t_i only using edges of multigraph $G(i)$, for each $i \in [k]$, where the total flow through any edge e is at most 1.

Intuitively, edges appear with priorities (think of the priority of an edge to be ℓ if it first appears in $G(\ell)$), and the request pair s_i-t_i can only use edges which have priority lesser than or equal to i to route its flow. It is easy to see that the maximum concurrent priority multicommodity flow can be computed efficiently using a linear programming formulation.

As for the dual notion of cuts, there are in fact two plausible definitions. One, inspired by the LP dual is what we call *priority cuts*, and the second, which will be easier to argue about, is what we call *prefix cuts*.

DEFINITION 4.2 (Priority Cuts). *Consider any sequence of multigraphs $G(1) \subseteq G(2) \subseteq \cdots \subseteq G(k)$ with unit edge-capacities and requests $\{(s_i, t_i, d_i) : i \in [k]\}$. We say that a set $Q \subseteq G(k)$ of edges priority separates pair i if and only if s_i and t_i are separated in the graph $G(i) \setminus Q$. The sparsity of a priority-cut Q is the ratio of $|Q|$ to the total demand of pairs that are priority separated by Q. The* **sparsest priority-cut** *is the minimum sparsity over all priority-cuts.*

DEFINITION 4.3 (Prefix Sparsity). *Consider any sequence of multigraphs $G(1) \subseteq G(2) \subseteq \cdots \subseteq G(k)$ with unit edge-capacities and requests $\{(s_i, t_i, d_i) : i \in [k]\}$. The prefix sparsity of this sequence is*

$$\min_{i=1}^{k} \quad \min_{S \subseteq V} \frac{|\delta_{G(i)}(S)|}{\sum_{1 \le j \le i : |S \cap \{s_j, t_j\}| = 1} d_j}.$$

4.1. Relationship between Prefix and Priority Sparsity. In this section, we relate the two definitions of sparsity. Indeed, from the definitions, it is clear that the value of the sparsest priority cut is at most the prefix sparsity, since every prefix cut is also a priority cut. However, the reverse direction is not obvious. Note that a demand j may be priority cut by some subset $Q \subseteq G(k)$ even though it is not separated in $G(i) \setminus Q$ for any $i > j$, i.e. j does not contribute to the i^{th} prefix-sparsity for $i > j$. To this end, we next show that there is indeed an approximate equivalence between the two notions of sparsity.

THEOREM 4.4. *Consider a sequence of unit-capacity multigraphs $G(1) \subseteq G(2) \subseteq \cdots \subseteq G(k)$ with requests $\{(s_i, t_i, d_i) : i \in [k]\}$. If the prefix-sparsity is at least α, then the sparsest priority cut is at least $\frac{\alpha}{2 \log k}$.*

Proof. Consider any $Q \subseteq G(k)$ that priority separates pairs $X \subseteq [k]$. For any subset $Y \subseteq [k]$ of requests, let $d(Y) = \sum_{i \in Y} d_i$ denote its total demand. We will show that $d(X) \le \frac{2 \log k}{\alpha} \cdot |Q|$, which would imply the desired lower bound on the sparsest priority-cut. Define graph $H(i) := G(i) \setminus Q$ for each $i \in [k]$. The proof is based on considering the connectivity structure in the sequence $H(1) \subseteq H(2) \subseteq \cdots \subseteq H(k)$. We say that at each *time* $i \in [k]$ the request-pair (s_i, t_i) arrives. At time j when (s_j, t_j) arrives, the edges $G(j) \setminus G(j-1) \setminus Q$ are added to graph $H(j-1)$ to get graph $H(j)$. Note that for each $i \in X$, the pair $s_i - t_i$ is separated in graph $H(i)$, by the definition of priority-cut Q.

To simplify the analysis, we assume (without loss of generality) that $H(k)$ has a single connected component: this can be ensured by adding a dummy request $k+1$ at the end where the newly arriving edges $G(k+1) \setminus G(k)$, which are disjoint from Q, contains a spanning tree.

For any subset of vertices $V' \subseteq V$, let $N(V') = d(\{i \in X : s_i, t_i \in V'\})$ be the total demand of requests in X that are induced in V'. We will show below by an inductive argument that $d(X) = N(V) \leq \frac{2\log k}{\alpha}|Q|$. In the following, we refer to the end points of request-pairs as terminals.

We now define a recurrence. Consider any $i \in [k]$ and a connected component C in $H(i)$. Let $j \leq i$ be the earliest time a request-pair arrived such that all vertices in C became connected in graph $H(j)$. Let $C_1, C_2, \ldots C_\ell$ be the components in $H(j-1)[C]$ which merged to become connected as C, at time j. By definition, $N(C_h)$ equals the total demand of pairs in X that are contained in C_h, for each $h \in [\ell]$. Note that $N(C)$ equals $\sum_{h=1}^{\ell} N(C_h) + d(I(C))$ where $I(C)$ denotes the set of requests in X "crossing" $\{C_h\}_{h=1}^{\ell}$, i.e. pairs having end points in two distinct components among $\{C_h\}_{h=1}^{\ell}$. For each $h \in [\ell]$ define:

- $Q_h = |\delta(C_h) \cap Q|$ the number of edges in Q with exactly one endpoint in C_h. Note that $Q_h = |\delta_{G(j-1)}(C_h)|$ because C_h is a connected component in $H(j-1) = G(j-1) \setminus Q$.
- $I_h = \{a \in X : a \leq j-1, |\{s_a, t_a\} \cap C_h| = 1\}$ the set of requests in X that arrive by time $j-1$ and have exactly one end point in C_h.

We index the components $\{C_h\}_{h=1}^{\ell}$ so that C_1 contains the maximum number of terminals. We claim that $I(C) \subseteq \bigcup_{h=2}^{\ell} I_h$. To see this, note that each request in $I(C)$ must have exactly one end-point in at least one component $\{C_h\}_{h=2}^{\ell}$. Moreover, $I(C) \subseteq [j-1]$ as each pair $b \in I(C)$ is in X and is induced on C which gets connected at time j: recall that s_b and t_b must be disconnected in graph $H(b)$. Thus we have

$$N(C) \quad \leq \quad \sum_{h=1}^{\ell} N(C_h) + \sum_{h=2}^{\ell} d(I_h).$$

We now use the prefix-sparsity condition to bound each $d(I_h)$. Consider the cut C_h in graph $G(j-1)$. The number of crossing edges $\left|\delta_{G(j-1)}(C_h)\right|$ is at most Q_h since C_h is a maximally connected component of $H(j-1) = G(j-1) \setminus Q$. Moreover, I_h is a subset of the requests with index at most $j-1$ crossing C_h. By the prefix-sparisty assumption, the sparsity of cut C_h in graph $G(j-1)$ is at least α, i.e.

$$\alpha \leq \frac{\left|\delta_{G(j-1)}(C_h)\right|}{d\left(\{a \in [j-1] : |C_h \cap \{s_a, t_a\}| = 1\}\right)} \leq \frac{Q_h}{d(I_h)}.$$

Combining the above two inequalities, we obtain

$$N(C) \quad \leq \quad \sum_{h=1}^{\ell} N(C_h) + \frac{1}{\alpha} \cdot \sum_{h=2}^{\ell} Q_h. \tag{4.1}$$

Consider expanding this recursion to obtain $N(V)$, which is possible as V is a connected component in $H(k)$. The base case of the recursion is singleton components, i.e. $N(\{v\}) = 0$ for any $v \in V$. Consider the contribution of each edge $e = (u, v) \in Q$ separately. Whenever e participates in the expression $\frac{1}{\alpha} \cdot \sum_{h=2}^{\ell} Q_h$ in (4.1), the number of terminals in the component containing either u or v doubles. This is because e must have one end-point in some $\{C_h\}_{h=2}^{\ell}$ and we chose indices such that $terminals(C_1) \geq terminals(C_h)$ for all $h \in [\ell]$. Thus, the number of times e contributes is at most $2\log_2 k$, and its total contribution is at most $\frac{2\log k}{\alpha}$. It follows that $N(V) \leq \frac{2\log k}{\alpha} \cdot |Q|$. This completes the proof. ∎

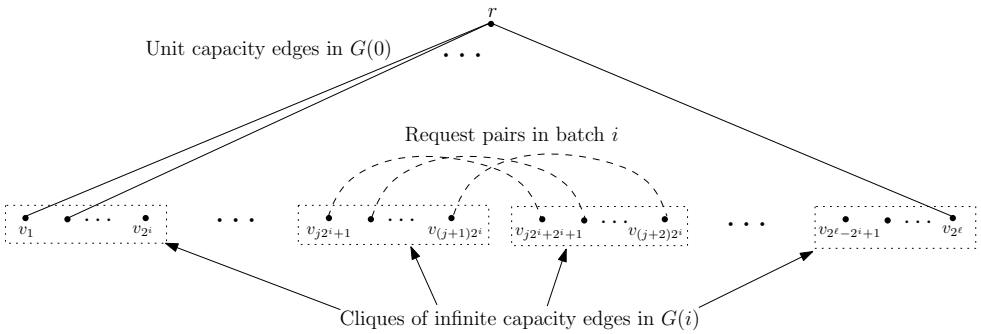

FIG. 1. *Gap instance for prefix sparsity and priority sparsest cut.*

Tight example. We now provide an example to show that the gap between prefix sparsity and priority sparsest cut can be $\Omega(\log k)$, thereby showing that Theorem 4.4 is tight up to constant factors. For clarity in exposition, we assume that request pairs arrive in batches of multiple request pairs per batch, and edges are also correspondingly added in batches.

Consider the following priority multigraph. There are $n = 2^\ell$ leaf vertices, say labeled $v_1, v_2, \ldots, v_n$, and one root vertex r. The graph $G(0)$ consists of unit-capacity edges (v_j, r) for $1 \le j \le n$. The graph $G(1)$ then consists of infinite capacity edges between $(v_1, v_2), (v_3, v_4), \ldots, (v_{n-1}, v_n)$. For each $1 \le i \le \ell$, graph $G(i)$ adds edges of infinite capacity between any pair of vertices in $\{v_{j\cdot 2^i+1}, v_{j\cdot 2^i+2}, \ldots, v_{(j+1)\cdot 2^i}\}$ for all $0 \le j \le 2^{\ell-i} - 1$. See Figure 1.

We now explain the request pairs which are introduced. In the batch corresponding to $G(0)$, for each $0 \le a \le 2^{\ell-1} - 1$, there is a request pair with unit demand introduced between v_{2a+1} and v_{2a+2}. For each $1 \le i \le \ell - 1$, the i^{th} batch corresponding to $G(i)$ has a request pair with unit demand between v_{j2^i+a} and $v_{(j+1)2^i+a}$ for all $1 \le a \le 2^i$ and all *even* $0 \le j \le 2^{\ell-i} - 1$. Note that the total number of request pairs is $k := \ell \cdot 2^{\ell-1}$ as the requests in each batch corresponds to a matching on $\{v_a\}_{a=1}^{2^\ell}$. See also Figure 1.

This completes the construction of our instance. To finish the analysis, we show that the priority sparsest cut value is at most $2/\ell$, and that the prefix sparsity is at least 1, thereby giving us a gap of $\ell = \Theta(\log k)$.

Indeed, note that the set of edges $Q = \{(v_j, r) : 1 \le j \le n\}$ forms a priority cut which priority separates all the request pairs. To see this, consider graph $G(i)$ for any $0 \le i \le \ell - 1$. Note that each edge in $G(i) \setminus Q$ is induced on some "group" $\{v_{j\cdot 2^i+1}, v_{j\cdot 2^i+2}, \ldots, v_{(j+1)\cdot 2^i}\}$ for $0 \le j \le 2^{\ell-i} - 1$. Whereas, every request pair in the i^{th} batch is between vertices from different groups. Hence, all the request pairs in the i^{th} batch (for $0 \le i \le \ell - 1$) are separated in $G(i) \setminus Q$, giving us the desired bound on the priority sparsest cut.

Next we note that the prefix sparsity is at least 1. Indeed, consider any i for $0 \le i \le \ell - 1$, and let us restrict our attention to the graph $G(i)$ and all the requests in the batches $0, 1, \ldots, i$. We now exhibit a concurrent multicommodity flow for these requests in $G(i)$. Indeed, for all request pairs in batches $0, 1, \ldots, i-1$, we directly send their flow using the infinite capacity edges present in $G(i)$. For each request (s, t) in the i^{th} batch, we route the flow along the $s - r - t$ path in $G(0)$. Note that these paths are edge disjoint as each vertex $\{v_a\}_{a=1}^{2^\ell}$ appears in exactly one request from

batch i. So all edge capacities are satisfied. As there is a feasible multicommodity flow, the prefix sparsity (in fact, even its LP relaxation) must be at least one.

4.2. Priority Multicommodity Flows and Priority Sparsity. We can express the maximum concurrent priority multicommodity flow problem as a linear program (see below). Let F^* denote the optimal value of this LP. The dual of this LP (which also has optimal value F^*) turns out to be a natural relaxation of the *sparsest priority cut* problem. To bound the flow-cut gap, we show that an optimal dual solution can be rounded to obtain an *integral priority cut* of sparsity $O(\log^2 k \cdot \log\log k)F^*$, which would establish an upper bound on the flow-cut gap for priority multicommodity flows. From a technical perspective, our rounding uses the region growing approach [22, 21] in a *recursive manner* to generate the priority cut, unlike traditional region-growing algorithms for sparsest cut. We remark that for the static case (without priorities), a better $\Theta(\log k)$ flow-cut gap is known; however, this relies on metric embedding ideas [34] which are not directly applicable in our setting with priorities. Improving our upper bounds (or obtaining better lower bounds) for the priority flow-cut gap is an interesting direction in its own right.

Priority Multicommodity Flow LP and Dual. We begin by stating the LP for priority multicommodity flow, and its dual problem. In the LPs below, $\mathcal{P}_i$ denotes the set of all s_i-t_i paths in $G(i)$. Also, in the primal formulation, the variable $f(p)$ denotes the amount of flow to be routed on some path p.

(PriorityFlowLP) $$\max \quad \gamma$$

(4.2) $$\text{s.t.} \sum_{p \in \mathcal{P}_i} f(p) \geq \gamma d_i \qquad \forall i \in [k]$$

(4.3) $$\sum_{p \mid e \in p} f(p) \leq 1 \qquad \forall e \in G(k)$$

(4.4) $$f(p) \geq 0 \qquad \forall i \in [k], \forall p \in \mathcal{P}_i$$

(PriorityCutLP) $$\min \sum_{e \in G(k)} z_e$$

(4.5) $$\text{s.t.} \sum_{i=1}^{k} d_i \eta_i \geq 1$$

(4.6) $$\sum_{e \in p} z_e \geq \eta_i \qquad \forall p \in \mathcal{P}_i \, \forall i \in [k]$$

(4.7) $$z_e \geq 0 \qquad \forall e \in G(k)$$

(4.8) $$\eta_i \geq 0 \qquad \forall i \in [k]$$

The feasible solutions for the primal LP are fractional routings such that each request-pair i routes at least a γ-fraction of its demand between them in graph $G(i)$ (constraint (4.2)), and such that no edge supports flow more than one (constraint (4.3)). This is precisely the priority multicommodity flow problem.

In the dual, we have an LP relaxation of the *sparsest priority-cut problem*: if an integral solution $Q \subseteq G(k)$ priority-cuts k' request-pairs, we set $\eta_i = 1/k'$ for the request-pairs which are priority separated, and $z_e = 1/k'$ for edges in Q and 0

otherwise. The objective value is then the sparsity of the priority-cut Q. We now present our main result of this section, which essentially establishes a bound on the flow-cut gap.

LEMMA 4.5. *Given a fractional solution to* (PriorityCutLP) *of value* F^**, we can obtain a priority cut of sparsity at most* $O(\log k \cdot \log(Dk) \cdot \log\log k)F^*$.

Here is an outline of the proof. Consider any fixed optimal solution (η^*, z^*) to (PriorityCutLP). First, we use a geometric scaling step (Claim 4.6) to reduce to a "priority multicut" problem where *all* requests in some subset must be priority cut. This step incurs an $O(\log kD)$-factor loss in the sparsity. Then we apply a variant of the region growing method (Lemma 4.5) to round fractional priority multicut solutions, which loses another $O(\log k \cdot \log\log k)$ factor.

We now define the priority multicut problem formally. An instance of priority multicut is given by a sequence $H(1) \subseteq H(2) \subseteq \cdots \subseteq H(r)$ of multigraphs with a set of demand pairs $\Pi = \{(s_i, t_i)\}_{i=1}^r$. The goal is to find a minimum size subset $Q \subseteq H(r)$ of edges that priority cuts each pair, i.e. $s_i - t_i$ is disconnected in $H(i) \setminus Q$ for all $i \in [r]$. The natural LP relaxation for priority multicut is:

$$\text{(MultiCutLP)} \qquad \min \sum_{e \in H(r)} z_e$$

$$\sum_{e \in p} z_e \geq 1 \qquad \forall i \in [r],\ \forall p \in \mathcal{P}_i, \tag{4.9}$$

$$z_e \geq 0, \qquad \forall e \in H(r) \qquad . \tag{4.10}$$

Above, $\mathcal{P}_i$ is the set of $s_i - t_i$ paths in graph $H(i)$.

CLAIM 4.6. *Given any solution* (η^*, z^*) *to* (PriorityCutLP)*, there is a subset* $\Pi \subseteq [k]$ *and solution* z *feasible to* (MultiCutLP) *for separating the requests in* Π *such that:*

$$\frac{\sum_e z_e}{\sum_{i \in \Pi} d_i} \leq 8 \log(Dk) \cdot \sum_e z_e^*.$$

Proof. For all $i \in [k]$ where $\eta_i^* \leq 1/(2Dk)$ we reset $\eta_i^* = 0$. Notice that since there are at most k variables η_i, this results in a solution to (PriorityCutLP) where the constraint (4.5) has

$$\sum_{i=1}^{k} d_i \cdot \eta_i^* \geq 1 - k \cdot D \cdot \frac{1}{2Dk} \geq \frac{1}{2}.$$

In other words, (4.5) is satisfied to extent at least $1/2$. We now geometrically group the η^* variables, according to classes

$$C_h = \{i \in [k] \mid 2^{-h} < \eta_i^* \leq 2^{-h+1}\}, \quad \text{for } h \in \{1, 2, \ldots, \log(2Dk)\}.$$

Let C_ℓ be the group that maximizes $\sum_{i \in C_\ell} d_i \eta_i^*$. Since there are at most $\log(2Dk)$ groups and $\sum_i d_i \eta_i^* \geq 1/2$, we have $2^{-\ell+1} \sum_{i \in C_\ell} d_i \geq \sum_{i \in C_\ell} d_i \eta_i^* \geq \frac{1}{(4 \log(Dk))}$, and so we have $\sum_{i \in C_\ell} d_i \geq 2^{\ell-3}/\log(Dk)$.

Now, to get our instance for priority multicut and associated fractional solution to (MultiCutLP), we simply set $\Pi = C_\ell$, i.e., all the requests in C_ℓ need to be priority cut. The graph sequence $G(1) \subseteq G(2) \subseteq \cdots G(k)$ is also restricted to the requests in Π. To ensure that every pair in C_ℓ is separated, we scale the metric $\{z_e^*\}_{e \in E}$ by a

factor of 2^ℓ, i.e., set $z_e = \min(1, 2^\ell z^*_e)$. It is now easy to see that z is a valid fractional solution for (MultiCutLP) for separating the requests in $\Pi = C_\ell$. Moreover,

$$\frac{\sum_e z_e}{\sum_{i\in\Pi} d_i} \le \frac{2^\ell \sum_e z^*_e}{2^{\ell-3}/\log(Dk)} \le 8\log(Dk)\cdot \sum_e z^*_e,$$

which proves the claim. ∎

We will use the fractional solution z to the priority multicut instance on Π (obtained from Claim 4.6) to find a priority cut of low sparsity. We restrict the graph sequence $G(1) \subseteq G(2) \subseteq \cdots G(k)$ to the requests in Π: let $H(1) \subseteq H(2) \subseteq \cdots H(r)$ denote this subsequence where $r = |\Pi|$. To reduce notation we also renumber the requests so that the requests in Π are numbered $1, 2, \cdots r$ in the order of their arrival.

We now proceed with our rounding algorithm. The next step relies on a variant of the region-growing technique [33, 22, 21]. Before describing the rounding, we introduce some useful notation. Given a graph $L \subseteq H(r)$, let d^L denote the shortest-path metric defined by $\{z_e : e \in L\}$, i.e. $d^L(u, v)$ is the length of the shortest path between u and v with weight z only on edges of L. For any vertex $v \in V$ and $\rho > 0$, define:

$$B^L(v,\rho) := \left\{u \in V : d^L(v,u) < \rho\right\} \text{ the } \textit{ball} \text{ of radius } \rho \text{ around } v \text{ in metric } d^L.$$

$$L(v,\rho) \text{ the induced graph of } L \text{ on vertices } B^L(v,\rho).$$

$$\delta^L(v,\rho) = \{(u,w) \in L : u \in B^L(v,\rho), w \notin B^L(v,\rho)\} \text{ the edges cut by } B^L(v,\rho).$$

$$\mathcal{V}^L(v,\rho) := \sum_{e\in L(v,\rho)} z_e + \sum_{(u,w)\in\delta^L(v,\rho)} \left(\rho - d^L(v,u)\right) + \frac{\mathcal{V}^*}{r}\cdot \mathit{terminals}\left(B^L(v,\rho)\right)$$

the *volume* of ball $B^L(v,\rho)$, where $\mathcal{V}^* = \sum_{e\in H(r)} z_e$ is the total LP volume.

We will use the following technical lemma:

LEMMA 4.7 ([21]). *For any $i \in [r]$ and $L \subseteq H(r)$ with $d^L(s_i, t_i) \ge 1$, there exists a value $0 < \rho < 1/2$ such that*

$$|\delta^L(s_i,\rho)| \quad \le \quad 4\log\log r \cdot \mathcal{V}^L(s_i,\rho)\cdot \log\left(\frac{2\cdot\mathcal{V}^L(s_i,1/2)}{\mathcal{V}^L(s_i,\rho)}\right).$$

In rounding solution z of (MultiCutLP), we will recursively generate the priority multicut Q. Recall that we have now restricted ourselves to the requests in Π, that are renumbered $[r] = \{1, 2, \cdots r\}$. The input to our recursive procedure is an index $i \in [r]$ and vertex subset $U \subseteq V$ such that i is the maximum index with both $s_i, t_i \in U$. Let Π_U denote the set of all requests in Π with both end-points in U. The initial call is with $i = r$ and $U = V$, and the solution $Q = \emptyset$ initially. For the recursive step, given i and U, we consider the induced graph $L = H(i)[U]$. Note that $d^{H(i)}(s_i, t_i) \ge 1$ by the feasibility of fractional solution z. Using the fact that both $s_i, t_i \in L \subseteq H(i)$ we have $d^L(s_i, t_i) \ge 1$. Let $Z := \sum_{e\in L} z_e + \frac{\mathcal{V}^*}{r}\cdot \mathit{terminals}(U)$ denote the volume of subgraph L. Applying Lemma 4.7 to both s_i and t_i, we find two radii $\rho_s, \rho_t < \frac{1}{2}$ such that:

$$|\delta^L(s_i,\rho_s)| \quad \le \quad 4\log\log r \cdot \mathcal{V}^L(s_i,\rho_s)\cdot \log\left(\frac{2\cdot Z}{\mathcal{V}^L(s_i,\rho_s)}\right), \quad \text{and}$$

$$|\delta^L(t_i,\rho_t)| \quad \le \quad 4\log\log r \cdot \mathcal{V}^L(t_i,\rho_t)\cdot \log\left(\frac{2\cdot Z}{\mathcal{V}^L(s_i,\rho)}\right).$$

Above we used the fact that $\mathcal{V}^L(s_i, 1/2) \leq Z$ since ball $B^L(s_i, 1/2) \subseteq L$; similarly $\mathcal{V}^L(t_i, 1/2) \leq Z$. Note that the balls $B^L(s_i, \rho_s)$ and $B^L(t_i, \rho_t)$ are disjoint as $d^L(s_i, t_i) \geq 1$ and $\rho_s, \rho_t < \frac{1}{2}$. So one of them, say the ball around s_i, has volume $X := \mathcal{V}^L(s_i, \rho_s) \leq Z/2$. We update the solution $Q \leftarrow Q \cup \delta^L(s_i, \rho_s)$. Let $U_1 \leftarrow B^L(s_i, \rho_s)$ and $U_2 \leftarrow U \setminus B^L(s_i, \rho_s)$. Note that all request-pairs $j \in \Pi_U$ which have exactly one end-point in U_1 are priority-cut by $\delta^L(s_i, \rho_s)$; in fact request j is separated even in graph $H(i) \supseteq H(j)$. The remaining pairs of Π_U are induced on either U_1 or U_2, and we handle them recursively in the two calls with U_1 and U_2 (along with the indices of the maximum induced requests). Let X' and Y denote the volumes of the induced graphs $H(i)[U_1]$ and $H(i)[U_2]$, on which we recurse.[2] Note that $X' \leq X$ and $Y \leq Z - X$.

In order to bound the total cost of our solution Q we make use of a recursive bound from [40, 21]. For any value $0 \leq x \leq \mathcal{V}^*$, let $f(x)$ denote the maximum cost of the priority cut computed by this procedure on any subgraph of volume x. Note that $f(x) = 0$ for $x < \frac{2\mathcal{V}^*}{r}$ since this would correspond to a subgraph with at most one terminal, i.e. containing no induced request-pair. We will show that $f(x) \leq 8 \log\log r \cdot x \cdot \log\left(\frac{2r \cdot x}{\mathcal{V}^*}\right)$. From the preceding discussion, for any volume Z subgraph we have:

$$f(Z) \quad \leq \quad \max_{\frac{\mathcal{V}^*}{r} \leq X \leq \frac{Z}{2}} \quad \left(4 \log\log r \cdot X \cdot \log \frac{2\,Z}{X} \;+\; f(X) \;+\; f(Z - X)\right).$$

Inductively substituting $f(X) \leq 8 \log\log r \cdot X \cdot \log\left(\frac{2rX}{\mathcal{V}^*}\right)$ and similarly for $f(Z - X)$, one can check directly (using $X \leq Z/2$) that the expression above is at most $8 \log\log r \cdot Z \cdot \log\left(\frac{2rZ}{\mathcal{V}^*}\right)$.

Clearly the total cost of Q is at most $f(3\mathcal{V}^*)$, which by the above calculation is $O(\log r \cdot \log\log r) \cdot \mathcal{V}^* = O(\log r \cdot \log\log r) \cdot \sum_{e \in H(r)} z_e$. Moreover, all demands in Π are separated. So the sparsity of priority cut Q is

$$O(\log r \cdot \log\log r) \cdot \frac{\sum_{e \in H(r)} z_e}{\sum_{i \in \Pi} d_i} \leq O(1) \log k \cdot \log(Dk) \cdot \log\log k \sum_e z^*_e,$$

where the inequality is by Claim 4.6. This completes the proof of Lemma 4.5.

An immediate consequence of Lemma 4.5 is:

THEOREM 4.8. *The worst-case ratio between sparsest priority-cut and maximum priority flow is $O(\log k \cdot \log(kD) \cdot \log\log k)$, where D is the max-to-min demand ratio.*

Proof. Given any instance $\mathcal{I}$ of priority multicommodity flow (Definition 4.1), let $\mathcal{I}'$ denote the instance with all demands and capacities scaled down by the minimum demand $\min_i d_i$. It is clear that the sparsest priority-cut remains the same in both $\mathcal{I}$ and $\mathcal{I}'$; a similar observation is true for the maximum priority flow. Note that the maximum demand in $\mathcal{I}'$ is $D = \max_i d_i / \min_i d_i$. We can now apply Lemma 4.5 to $\mathcal{I}'$ to obtain an $O(\log k \cdot \log(kD) \cdot \log\log k)$ ratio between the priority cut and flow. So we obtain the same ratio for instance $\mathcal{I}$ also. ∎

4.3. Removing Dependence on Demand Values. We now show that the dependence on the maximum demand can be eliminated in Theorem 4.8. The only part in our analysis which has the dependence on D is in Claim 4.6, which reduces

[2] The actual volumes may be even smaller as the recursive call on U_1 (similarly U_2) has volume $H(j)[U_1]$ where $j < i$ is the maximum index with $s_j, t_j \in U_1$.

priority sparsest cut to multicut. Our idea to eliminate the dependence on D is to (i) break up the original instance into several sub-instances, such that the ratio of max-to-min demand is polynomial in k in each sub-instance, and (ii) combine the flow solutions across all subinstances while incurring only an extra constant-factor in congestion. This high-level approach is similar to that of Plotkin and Tardos [36] for flow-cut gaps in the static setting where earlier bounds of $O(\log k \log D)$ [31] were improved to $O(\log^2 k)$. However, the specific flow combination used in [36] is inapplicable in the priority setting. So we provide a different method below.

THEOREM 4.9. *The worst-case ratio between sparsest priority-cut and maximum priority flow is $O(\log k \cdot \log n \cdot \log\log k)$, where n is the number of vertices.*

Proof. By assigning edge capacities, we assume (without loss of generality) that the number of edges in each graph $G(i)$ is at most $m = O(n^2)$. Using LP extreme point properties, it follows that any flow for any request $i \in [k]$ can be decomposed into at most m non-zero flow paths.

As a first step, we break up the demands into classes where

$$\mathcal{D}_h = \{i \in [k] : (2m)^{4h} \le d_i < (2m)^{4(h+1)}\}, \qquad \forall h \in \mathbb{Z}.$$

Note that the ratio of maximum demand to minimum demand in each class is at most m^4. By Theorem 4.8, the flow-cut ratio for all requests within any class is $\rho := O(\log k \cdot \log n \cdot \log\log k)$. Hence there is a priority multicommodity flow $\mathcal{F}_h$ routing all demands in each class h with congestion ρ.

It suffices to obtain a routing of all the demands with congestion $O(\rho)$. We combine flows in the following manner. We combine all the odd classes into one flow of $O(\rho)$ congestion, and all the even classes into another flow of $O(\rho)$ congestion. Putting the two together would imply the desired flow of $O(\rho)$ congestion for all classes.

To this end, we now show how to combine the flows of the odd classes into one common flow (handling even classes is identical). Here, we use the fact that the number of non-zero flow paths for each request in class $\mathcal{F}_h$ is at most m. We simplify each flow $\mathcal{F}_h$ in the following manner: for each $i \in \mathcal{D}_h$, simply ignore all flow paths carrying less than $d_i/(2m)$ flow. Note that after this, the total flow routed for each demand i is still at least $d_i/2$. Moreover, each edge carrying non-zero flow in $\mathcal{F}_h$ in fact carries a flow of at least $(2m)^{4h}/(2m)$. Now we observe that *even if we simply add all flows* into a single flow, the overhead in congestion would be small. Formally, consider any edge e and let h denote the largest class whose demands route flow using e. Then, the total utilization of edge e by demands of all odd classes *before* h is $L_e^{<h} \le k \cdot m^{4(h-1)} \le m^{4h-3}$. (Note that as we are only combining odd classes, all demands from previous classes are at most $(2m)^{4(h-1)}$.) On the other hand, the utilization of e in just $\mathcal{F}_h$ is at least $(2m)^{4h-1} \ge \frac{1}{2} \cdot L_e^{<h}$. So we can combine all odd classes with only a factor 4 increase in congestion. ∎

5. Online Algorithm. In this section we provide a randomized online algorithm for EERP with competitive ratio $O(\log^{3\alpha+1} k \cdot (\log\log k)^{2\alpha})$, which proves Theorem 1.2. For better clarity in presentation, we will first assume that k is known up front to the algorithm, and in Section 5.2 we discharge this assumption to make the algorithm truly online. Recall that we use $\lambda = O(\log k)$ as in the offline algorithm; this value is determined since we assume we know the value of k.

Similar to the offline algorithm, our overall online algorithm comprises two stages: in the first "buying" stage, we decide which edges to power on (or buy) and incur the

static cost, in an online manner. Once this is done, the natural thing to do would be to route all the requests using the edges bought, to minimize the dynamic power of the routing. Note that the waterfilling algorithm is naturally suited for solving this DynPM problem in an online manner. We perform the buying stage in steps 1 and 2 below, and the final routing in step 3 below.

Our algorithm separately runs two instances of the online waterfilling algorithm (Theorem 2.1). The first one routes the hallucinated demands to determine the edges to buy, and the second one actually routes all the original demands. The first instance is called $\mathcal{I}_{hal}$ and the second is called $\mathcal{I}_{act}$. Initially, both are empty instances.

In response to request-pair (s_i, t_i) the algorithm takes the following steps:
1. Augmenting the Steiner backbone: We run an online algorithm for Steiner forest to connect s_i and t_i, where the edges have costs $\{c_e : e \in E\}$.
2. Augmenting the Hallucination backbone: Request $i \in [k]$ "hallucinates" a demand of $q \cdot B_i$ units, where $B_i \sim \mathsf{Binomial}(d_i, \frac{\lambda}{q})$. We feed the hallucinated demand to the online waterfilling algorithm (Theorem 2.1) for instance $\mathcal{I}_{hal}$, which updates its routing $\mathcal{H}$ to add an unsplittable routing of $q \cdot B_i$ units of flow between s_i and t_i. Let $G_H(i) \supseteq G_H(i-1)$ denote the subgraph of all edges used in $\mathcal{H}$.
3. Routing: Define graph $G_F(i) := G_S(i) \cup G_H(i)$, and let $\mathcal{P}_i$ denote the set of all s_i-t_i paths in $G_F(i)$. Now feed the actual demand (s_i, t_i, d_i) to the online waterfilling algorithm (Theorem 2.1) for instance $\mathcal{I}_{act}$ to find a path $P_i \in \mathcal{P}_i$. Output P_i as the routing of request i.

5.1. Analysis. The outline of the analysis closely mirrors that of Section 3, with the main difference being that we use the bounds established in Section 4 on the flow-cut gap of priority multicommodity flows instead of the classical flow-cut gaps we used in Section 3. We now show that our online algorithm, which knows k upfront, is $O\left(\log^{\alpha+3} k \cdot (\log\log k)^{\alpha}\right)$-competitive for EERP. The high-level approach is the same as for the offline problem.

Static power of $G_S(k)$. To maintain an online Steiner forest, we use the $O(\log k)$-competitive algorithm from [12]. This allows us to bound the static power of edges in $G_S(k)$ by $O(\log k) \cdot \mathsf{Opt}$.

Static power of $G_H(k)$. We bound the static power for the hallucination backbone by its dynamic power (as each hallucinated demand is at least q), which in turn can be bounded by $O(\log^{\alpha} k) \cdot \mathsf{Opt}$, using an argument analogous to that of Lemma 3.1.

Dynamic power. The analysis of the dynamic power of the algorithm's routing is significantly more involved than in the offline case. Again, using Theorem 2.1, it suffices to prove the *existence* of a routing of low dynamic power where each request i is routed on a path from $\mathcal{P}_i$, which is the set of all s_i-t_i paths in the current graph $G_F(i)$. But how do we ensure that there exists a good (offline) routing with the additional restriction on the edges allowed for each request? In the offline algorithm, because the backbone graph was static, we could appeal to the classic multicommodity flow-cut gap: this is no longer applicable in the online case. Indeed, this is the main difference from the analysis of the offline algorithm, and we resort to the bounds we established on the flow-cut gap of priority multicommodity flows in Theorem 4.9.

LEMMA 5.1. *The expected static power of the backbone $G_F(k)$ is $O(\lambda^{\alpha}) \cdot \mathsf{Opt}$.*

Proof. The proof is identical to Lemma 3.2 in the offline case, except that the static power of $G_S(k)$ is $O(\log k) \cdot \mathsf{Opt}$ due to the online Steiner forest algorithm. ∎

In order to show that the dynamic power of our routing is small, we need to show that there is a low-congestion *priority* multicommodity flow on the backbone edges.

Formally, let $\{H_e(i) : e \in E\}$ denote the edge loads induced by the routing $\mathcal{H}$ after arrival of request pair s_i-t_i. Then we define the following edge capacities:

$$\widehat{q}_e(i) = \begin{cases} H_e(i) & \text{if } e \in G_F(i) \setminus G_S(i) \\ H_e(i) + \lambda q & \text{if } e \in G_S(i) \end{cases}, \qquad \forall e \in E.$$

Note that only edges in $G_F(i) = G_S(i) \cup G_H(i)$ have nonzero capacity. Also, note that the values $\widehat{q}_e(i)$ are monotonically non-decreasing over the value of i, since the routing $\mathcal{H}$ is an online routing which doesn't re-route any flows, and G_S is the output of an online Steiner forest algorithm which only adds edges. Finally, similar to the offline algorithm, we can bound $\widehat{q}_e(k)^\alpha \le 2^\alpha \left(H_e(k)^\alpha + \lambda^\alpha \sigma \cdot \mathbf{1}_{e \in G_S(k)}\right)$, so we have:

$$\sum_{e \in E} c_e \mathbb{E}[\widehat{q}_e(k)^\alpha] \le 2^\alpha \left(\sum_{e \in E} c_e \mathbb{E}[H_e(k)^\alpha] + \sigma \lambda^\alpha \cdot \sum_{e \in G_S(k)} c_e \right) = O(\lambda^{\alpha+1}) \cdot \mathsf{Opt}, \tag{5.1}$$

where the last inequality uses (i) the static power of $G_S(k)$ is $\sigma \cdot \sum_{e \in G_S(k)} c_e \le O(\log k) \cdot \mathsf{Opt}$ due to the online Steiner forest algorithm and (ii) the expected dynamic power of $\mathcal{H}$ is $\sum_{e \in E} c_e \mathbb{E}[H_e(k)^\alpha] = O(\lambda^\alpha) \cdot \mathsf{Opt}$ by Lemma 3.1.

Equipped with the above notations, we show that the prefix-sparsity of the graphs $G_F(1) \subseteq G_F(2) \subseteq \cdots \subseteq G_F(k)$ is large w.r.t the original demands, where an edge $e \in G_F(i)$ has capacity $\widehat{q}_e(i)$. This is analogous to Lemma 3.3 for the offline setting.

LEMMA 5.2. *With probability at least $1 - O(1/k^{2\alpha})$, the prefix sparsity of the sequence of graphs $G_F(i)$ (equipped with edge capacities $\{\widehat{q}_e(i)\}$) is at least $\lambda/3$.*

Proof. Fix some i. Since all pairs in $[i] = \{1, 2, \cdots i\}$ have hallucinated independently, we can apply Lemma 3.3 and conclude that the sparsity of $G(i)$ is at least $\lambda/3$ with requests $[i]$. The lemma then follows by a simple union bound over all $i \in [k]$. ∎

COROLLARY 5.3. *With probability at least $1 - k^{-2\alpha}$, there exists a fractional priority routing that respects capacities $O(\log^2 k \cdot \log\log k) \times \widehat{q}$.*

Proof. We first argue that it suffices to exhibit a good priority multicommodity flow in the following sequence of virtual graphs $\mathcal{B}(1) \subseteq \mathcal{B}(2) \subseteq \ldots \subseteq \mathcal{B}(k)$, with $\mathcal{B}(i)$ defined on vertices V with the following edges and capacities:

- *Steiner* edges: each edge $e \in G_S(i)$ has capacity $\bar{q}_e = \lambda q$ in $\mathcal{B}(i)$.
- *Hallucinated* edges: for each $j \in [i]$ we add a direct edge (s_j, t_j) with capacity $\bar{q}_{(s_j,t_j)} = q \cdot B_j$ in $\mathcal{B}(i)$.

Note that each hallucinated edge (s_j, t_j) in $\mathcal{B}(i)$ corresponds to an actual $s_j - t_j$ path carrying $q \cdot B_j$ flow in the actual hallucinated routing $\mathcal{H}$ (based on which we assign the virtual capacities $\widehat{q}_e$ in the first place). Hence, if we find a low congestion priority multicommodity flow in $\mathcal{B}$, then it is easy to translate it into a low congestion priority flow in G_F with virtual capacities $\widehat{q}_e$. Finally, we make another simplification which again does not alter the flows: for any degree two Steiner vertex v in $\mathcal{B}(k)$ where both its incident edges (u, v) and (v, w) are from the Steiner forest $G_S(k)$, short-cut these edges, i.e. remove (u, v) and (v, w), and add edge (u, w) with capacity λq. By repeatedly doing this, we will end up with a graph without such degree two Steiner vertices, which will imply that the total number of edges in $\mathcal{B}(k)$ (and hence in all its subgraphs) is at most $O(k)$. Now we can utilize Theorem 4.9 to obtain an $O(\log^2 k \cdot \log\log k)$ flow-cut gap for such instances. Combining this fact with Lemma 5.2 and Theorem 4.4 then completes the proof. ∎

LEMMA 5.4. *The expected dynamic power of the online algorithm's routing is*

$$O(1) \times \left(\mathbb{E}[\textit{Static power of } G_F] \; + \; \rho^\alpha \sum_e c_e \mathbb{E}[\widehat{q}_e^\alpha] \right) \; \leq \; O(\lambda^{\alpha+1}\, \rho^\alpha) \cdot \mathsf{Opt},$$

where $\rho = O(\log^2 k \cdot (\log\log k))$.

Proof. This proof is identical to Lemma 3.5 (for the offline problem), where we:

- use Corollary 5.3 to get the existence of a fractional routing respecting capacities $\rho \cdot \widehat{q}$ (instead of Corollary 3.4),
- use Lemma 5.1 to bound the static power (instead of Lemma 3.2),
- use (5.1) to bound $\mathbb{E}[\widehat{q}_e^\alpha]$ (instead of (3.1)), and
- use the online guarantee from Theorem 2.1 for minimizing dynamic power. ∎

Combining Lemmas 5.1 and 5.4, we obtain:

THEOREM 5.5. *There is an* $O(\log^{3\alpha+1} k \cdot (\log\log k)^\alpha)$*-competitive randomized online algorithm for* EERP *when the number* k *of requests is known.*

5.2. When Number of Requests is Unknown. Our algorithm extends easily to the truly online setting when the final number k of request-pairs is not known in advance. The only place where our online algorithm relies on knowledge of k is in the hallucination step. Recall that each request i hallucinates a demand of $q \cdot \mathsf{Binomial}(d_i, \lambda/q)$ units, where the parameter $\lambda = O(\log k)$. Let $\lambda(i) = O(\log i)$ where the constant factor in the big-O is the same as for λ. We now modify the online algorithm as follows. When request-pair i arrives, we hallucinate a demand of $q \cdot \mathsf{Binomial}(d_i, \lambda(i)/q)$ units. Subsequently, upon the arrival of each request $j \geq i$, we ensure that request-pair i has total hallucinated demand of $q \cdot \mathsf{Binomial}(d_i, \lambda(j)/q)$ units. This can be done easily by re-sampling i's hallucinated demand (with the appropriate probability) after each request arrival $j > i$.

We will show that the competitive ratio of this algorithm is only an $O((\log\log k)^\alpha)$ factor more than that in Theorem 5.5. Recall (from Section 5.1) the definitions of capacities $\widehat{q}$ and graphs $G_F(1) \subseteq G_F(1) \subseteq \cdots G_F(k)$. We continue to have Lemma 5.1 and (5.1). However, the prefix sparsity condition (Lemma 5.2) is no longer true, as our sampling probabilities are now smaller. We have the following (weaker) version of Lemma 5.2 (the proof is identical).

LEMMA 5.6. *For each* $i \in [k]$*, with probability at least* $1 - O(1/i^{2\alpha})$*, the sparsity of multigraph* $G(i)$ *with demands* $\{(s_j, t_j)\}_{j=1}^i$ *is at least* $\Omega(\log i)$.

We first analyze the algorithm assuming the following.

(5.2) Assume that for each $i \in [k]$, the sparsity of $G(i)$ is at least $\log i$.

Later we show how to handle the (low probability) case where (5.2) does not hold.

By (5.2), the prefix-sparsity of the sequence $G_F(1) \subseteq G_F(2) \subseteq \cdots G(k)$ is $\Omega(1)$. Using this in Theorem 4.4 (instead of the $\log k$ prefix-sparsity from Lemma 5.2) we immediately obtain that the sparsest priority-cut is $\Omega(1/\log k)$. We can actually obtain a better bound by modifying the proof of Theorem 4.4, as shown below.

THEOREM 5.7. *Consider a sequence of multigraphs* $G(1) \subseteq G(2) \subseteq \cdots G(k)$ *with requests* $\{(s_i, t_i, d_i) : i \in [k]\}$*. Assume that for each* $i \in [k]$*, the sparsity of multigraph* $G(i)$ *with requests indexed* $\{1, 2, \cdots i\}$ *is at least* $\log i$*. Then the sparsest priority cut is* $\Omega\left(\frac{1}{\log\log k}\right)$.

Proof. We only provide an outline as the proof is almost identical to that of Theorem 4.4. The difference is in inequality (4.1) which now becomes:

$$N(C) \quad \leq \quad \sum_{h=1}^{\ell} N(C_h) + \frac{1}{\log j} \cdot \sum_{h=2}^{\ell} Q_h. \tag{5.3}$$

Recall that here C is a connected component that forms at time j (i.e. when request j arrives) due to the merging of components $\{C_h\}_{h=1}^{\ell}$ in graph $H(j-1)$. Also C_1 is the component in $\{C_h\}_{h=1}^{\ell}$ with the maximum number of terminals. This change in the recurrence for $N(\cdot)$ affects the total contribution of each edge $e = (u, v) \in Q$ in the expansion of $\sum_{D:comp(H(k))} N(D)$. Recall that whenever e contributes to this sum, the number of terminals in the component containing either u or v doubles. Consider the total contribution of e due to components containing u (the case of v is identical). Let $2 \leq T_1 \leq T_2 \leq \cdots \leq T_\beta \leq 2k$ denote the number of terminals in u's component whenever e contributes due to u. Recall that the number of terminals in u's component at least doubles each time, i.e. $T_{b+1} \geq 2 \cdot T_b$ for all $1 \leq b < \beta$. Also, note that if u's component has T_b terminals then the total number of terminals at that time $j_b \geq T_b$: so e's contribution at this time is $\frac{1}{\log j_b} \leq \frac{1}{\log T_b}$. Therefore, e's total contribution is at most:

$$\sum_{b=1}^{\beta} \frac{1}{\log j_b} \quad \leq \quad \sum_{b=1}^{\beta} \frac{1}{\log T_b} \quad \leq \quad \sum_{b=1}^{\beta} \frac{1}{\log (2^b)} \quad = \quad \sum_{b=1}^{\beta} \frac{1}{b} \quad \leq \quad \log(2 \log k).$$

This completes the proof of Theorem 5.7. ∎

Exactly as in Corollary 5.3, we obtain:

Corollary 5.8. *Assuming (5.2), there exists a fractional priority routing that respects capacities $O(\log^2 k \cdot \log\log k) \times \widehat{q}$.*

The proof is identical to Corollary 5.3, where we use Theorem 5.7 (instead of Theorem 4.4) along with the flow-cut gap (Theorem 4.9).

We now bound the overall expected dynamic power.

Lemma 5.9. *The expected dynamic power of the online algorithm's routing is*

$$O(1) \times \left(\mathbb{E}[\textit{Static power of } G_F] \; + \; \rho^\alpha \sum_e c_e \mathbb{E}[\widehat{q}_e^\alpha] \right) \; \leq \; O(\lambda^{\alpha+1} \rho^\alpha) \cdot \mathsf{Opt},$$

where $\rho = O(\log^2 k \cdot (\log\log k)^2)$.

Proof. This proof is very similar to Lemmas 5.4 and 3.5. Let

$$I = \{i \in [k] : \text{ graph } G(i) \text{ has sparsity at least } \log i\}.$$

Then, if we restrict to the requests in I then (5.2) is satisfied. Now, exactly as in Lemma 5.4 (using Corollary 5.8, Lemma 5.1, Inequality (5.1) and Theorem 2.1), the expected dynamic power to route I is $O(\lambda^{\alpha+1} \rho^\alpha) \cdot \mathsf{Opt}$.

Now we bound the expected cost due to requests $\bar{I} = [k] \setminus I$. We will bound the optimal dynamic power of instance $\mathcal{I}_{act}$ restricted to $\bar{I}$. To this end, consider the feasible routing that sends d_i units along the unique $s_i - t_i$ path in G_S for each $i \in \bar{I}$. The expected dynamic cost of this routing is at most:

$$\left(\sum_{e \in G_S} c_e \right) \cdot \sum_{i=1}^{k} \Pr[i \in \bar{I}] \cdot (iq)^\alpha \leq \left(\sum_{e \in G_S} c_e \right) \cdot \sum_{i=1}^{k} i^{-2\alpha} \cdot (iq)^\alpha \leq O(1) \cdot \sigma \cdot \sum_{e \in G_S} c_e,$$

where the first inequality uses Lemma 5.6. So the expected optimal dynamic power of $\mathcal{I}_{act}$ restricted to $\bar{I}$ is at most $O(1)$ times the static power of G_S. Using Theorem 2.1, the expected dynamic power of our online algorithm on $\bar{I}$ is also $O(1)$ times the static power of G_S.

Combining the costs due to requests in I and $\bar{I}$, the lemma follows. ∎

Using Lemmas 5.1 and 5.9, we obtain:

THEOREM 5.10. *There is an $O(\log^{3\alpha+1} k \cdot (\log\log k)^{2\alpha})$-competitive randomized online algorithm for EERP.*

6. Multicommodity Capacitated Network Design. In this section, we consider the (uniform) capacitated multicommodity network design problem (CapND) as studied by [3]. The CapND problem is also called the *fixed-charge network design* problem in the operations research literature. In uniform CapND, we are given an undirected multigraph $G = (V, E)$ with each edge $e \in E$ having a cost c_e and capacity $q \geq 0$ (same across all edges). We are also given a collection of k request-pairs $\{(s_i, t_i) : i \in [k]\}$ each with demand $d_i \geq 0$. The goal is to choose a minimum cost subgraph $H \subseteq G$ such that H can support a concurrent multicommodity flow of the request-pairs. Let $m = |E|$ denote the number of edges in G. Unlike the earlier sections, the flow for each request does not have to be *unsplittable*. In the case that each demand is at most q, we will in fact see that our algorithm guarantees an unsplittable routing even if the optimum is splittable. We assume (without loss of generality) that there is at most one request between each pair of vertices, so $k \leq |V|^2 \leq m^2$.

We only consider the online version of this problem where the requests arrive over time, and one needs to buy edges in an online fashion so that the current set of edges can support the desired multicommodity flow. We will prove Theorem 1.3, i.e. an $(O(\log m), O(\log^2 m))$ bicriteria competitive ratio. An (α, β) bicriteria performance guarantee means that the solution has (i) cost at most α times the optimum (with edge capacities q) and (ii) the total flow on each chosen edge is at most $\beta \cdot q$. For the (simpler) offline version, we can obtain a slightly better $(O(\log m), O(\log m))$ bicriteria approximation algorithm as described in the conference version of this paper [5]: we do not discuss this result here.

At a high level, our algorithm is similar to those in Sections 3 and 5. The difference is that we now try to minimize congestion (which corresponds to the maximum load on edges) rather than the energy cost (which corresponds to the sum of α powers of these loads). We first consider (in Section 6.1) the special case when each demand is at least the capacity q. We then reduce the general case to this special case using the hallucination idea (in Section 6.2).

6.1. Online CapND For Demands At Least q. As a first step, note that if each demand is at least q, then we can ensure that each demand d_i is an integer multiple of q by considering demand $\lceil \frac{d_i}{q} \rceil \cdot q$ instead: this only loses a constant factor in the congestion bound. Further, each request (s_i, t_i, d_i) can be split into $\lceil \frac{d_i}{q} \rceil$ many requests of demand q each with the same terminals s_i and t_i. Hence, for the remainder of the subsection, we assume that all demands are actually *equal* to q, the common edge capacity. Note that we may assume that the total demand $\sum_{i=1}^{k} d_i \leq mq$, as otherwise the CapND instance must be infeasible (the total available capacity is mq). This means that the number of demand-q requests after the above modification is at most $k + m$, which is polynomial. The requests (s_i, t_i) arrive online, and the algorithm must irrevocably buy edges so that, at any point, the subgraph bought by the algorithm can simultaneously support a (splittable) multi-commodity flow of q

demands for each request. We refer to this special case of CapND as $\mathsf{CapND_q}$.

A closely related problem is *min-cost circuit routing* that was considered in [9]. Here, the routing for each request is *unsplittable* (rather than splittable) and the cost of the solution is the *sum* of costs over all paths used in the routing.

THEOREM 6.1 ([9]). *There is an* $(O(1), O(\log C))$ *bicriteria competitive algorithm for min-cost circuit routing, where* $C = \sum_{e \in E} c_e$ *is the total cost of edges.*

We note that this result also holds when we compare to the *splittable* optimal solution to min-cost circuit routing: the proof from [9] extends immediately by averaging over all the flow paths in the splittable optimum. Moreover, for the $\mathsf{CapND_q}$ problem (where all demands and capacities are q), the cost guarantees carry over directly from the min-cost circuit routing problem: so we obtain an $(O(1), O(\log C))$ bicriteria competitive algorithm for $\mathsf{CapND_q}$. We will refer to this algorithm as $\overline{\mathcal{A}}_{\mathsf{CND}}$.

We can obtain an improved $(O(1), O(\log m))$ bicriteria guarantee for $\mathsf{CapND_q}$ using a standard guess-and-double approach. At any point let B denote an upper bound on the optimal cost (initially this equals the minimum edge cost). Then, we simply ignore edges of cost more than B, and update the cost of any edge e with cost $c_e \leq B/m$ to be equal to $\hat{c}_e = B/m$. Note that for the modified instance, all edge costs $\hat{c}$ vary between B/m and B: so we may assume (by scaling) that the maximum edge cost is m, which implies that $C \leq m^2$. We then pass the modified instance to algorithm $\overline{\mathcal{A}}_{\mathsf{CND}}$ and return its output. We double the guess B whenever algorithm $\overline{\mathcal{A}}_{\mathsf{CND}}$ "fails", i.e. either its cost exceeds $O(1) \cdot B$ or the congestion exceeds $O(\log m)$. Whenever B is doubled, we run the algorithm from Theorem 6.1 on the entire input sequence again (this time with the new value of B). Note that this is still a valid online algorithm for $\mathsf{CapND_q}$ as we do not need to commit to routing paths in an online fashion. Clearly, the total cost incurred is $O(1)$ times the optimal cost of $\mathsf{CapND_q}$. By considering only the edges bought in the last run of the algorithm from Theorem 6.1, it is clear that there exists a routing for all requests with congestion $O(\log m)$. So we obtain:

LEMMA 6.2. *There is an an* $(O(1), O(\log m))$ *bicriteria competitive algorithm for uniform capacitated network design when all demands are at least* q*.*

We will refer to this algorithm as $\mathcal{A}_{\mathsf{CND}}$.

6.2. Online CapND. We now consider the general demands setting. As a first step, we split the instance into two sub-instances, one which handles all requests (s_i, t_i, d_i) such that $d_i \geq q$, and the other which handles the requests (s_i, t_i, d_i) such that $d_i < q$. If we separately solve each of the sub-instances and combine them, the total cost and congestion would just add up. So this splitting only incurs a factor 2 overhead. We can solve the first sub-instance (demands at least q) using algorithm $\mathcal{A}_{\mathsf{CND}}$. For the remainder of this section, we assume that all demands are at most q. We handle this case by using our hallucination idea to *randomly scale up* some demands to integer multiples of q, after which we can use algorithm $\mathcal{A}_{\mathsf{CND}}$.

Online CapND Algorithm when all demands at most q.
1. Constructing the Steiner backbone $G_S(i)$. Solve the online Steiner forest instance on graph $G = (V, E)$ with pairs $\{(s_i, t_i)\}_{i=1}^{k}$ using the $O(\log k)$-competitive online algorithm from [12]. Let $G_S(i) \supseteq G_S(i-1)$ denote the resulting solution, maintained incrementally.
2. Constructing the Hallucination backbone $G_H(i)$. Request $i \in [k]$ "hallucinates" a demand of $q \cdot B_i$ units, where $B_i \sim \mathsf{Binomial}(d_i, \frac{\lambda}{q})$ and $\lambda = \Theta(\log k)$. Let $\widetilde{G}$ denote the network obtained from G by replacing each edge $e \in E$ by $\rho = O(\log m)$ many parallel "copies", each of cost c_e and capacity q. Feed the hallucinated demand in network $\widetilde{G}$ to the online algorithm $\mathcal{A}_{\mathsf{CND}}$, which maintains subgraph $\widetilde{G_H}(i)$ incrementally. Let $G_H(i)$ denote the subgraph of G that contains an edge $e \in E$ if and only if $\widetilde{G_H}(i)$ contains any copy of e.
3. Output. Return $G_S(i) \cup G_H(i)$ as the final solution.

6.3. Analysis. The analysis proceeds along the same lines as in Section 3. We prove the following lemmas, the combination of which will prove Theorem 1.3.

LEMMA 6.3. *The cost of the Steiner forest G_S is at most $O(\log k) \cdot \mathsf{Opt}$.*

Proof. Since the optimal solution supports a multicommodity flow between all the request pairs, it contains a Steiner forest connecting each s_i and t_i. The lemma now follows as we use an $O(\log k)$-competitive algorithm for Steiner forest [12]. ∎

LEMMA 6.4. *With high probability, graph $G_H(k)$ has cost $O(\log m)\mathsf{Opt}$ and can route all the hallucinated demands with edge congestion $O(\log^2 m)$.*

Proof. The proof is similar to that of Lemma 3.1, except that we now need to bound the maximum load on any edge as opposed to the total α^{th} power of the loads. Consider the optimal CapND solution along with a splittable routing $\mathcal{S}_i$ for each request i; note that the total load on any edge in this solution is at most q. Let $\mathcal{O}$ denote the following random routing: for each request i, sample B_i many $s_i - t_i$ paths from $\mathcal{S}_i$ and send demand of q along each of these paths. We will show that $\mathcal{O}$ corresponds to a solution to the hallucinated CapND instance with cost at most Opt and congestion at most $\rho = O(\log m)$.

Since we only use edges of the optimal CapND solution, the cost of edges used by $\mathcal{O}$ is at most Opt. Moreover, the load on any edge (under the routing $\mathcal{O}$) is the sum of independent Bernoulli random variables (scaled by q) with mean at most $\lambda q = O(\log m) \cdot q$. So a straightforward application of Chernoff bounds implies that the maximum load on any edge is at most $O(\log m) \cdot q$ with high probability. Consider now the following solution to the hallucinated CapND instance on network $\widetilde{G}$. For all $e \in E$, if $\mathcal{O}$ sends $\theta \cdot q$ flow through e then the solution contains θ copies of edge e. Therefore, the optimal value of the hallucinated CapND instance is at most $\rho \cdot \mathsf{Opt}$.

Lemma 6.2 now implies that algorithm $\mathcal{A}_{\mathsf{CND}}$ obtains (with high probability) a solution to the hallucinated instance (in $\widetilde{G}$) with cost $O(\rho) \cdot \mathsf{Opt}$ and congestion $O(\log m)$. This translates to a solution in the original network G with cost $O(\log m) \cdot \mathsf{Opt}$ and congestion $O(\rho \log m) = O(\log^2 m)$. ∎

LEMMA 6.5. *With high probability, graph $G_S(k) \cup G_H(k)$ can support routing all demands with congestion $O(\log^2 m)$.*

Proof. Much like the proof of Lemma 3.3, we consider a *virtual* graph $\mathcal{B}$ on vertices V with the following edges and capacities:

- *Steiner* edges: each edge $e \in G_S(k)$ has capacity $\bar{q}_e = \lambda q$.

- *Hallucinated* edges: for each $i \in [k]$ edge (s_i, t_i) has capacity equal to its hallucinated demand, i.e., $\bar{q}_{(s_i,t_i)} = q \cdot B_i$.

Using an argument identical to that of Lemma 3.3, we can show that the sparsity of $\mathcal{B}$ with respect to the original demands is at least $\lambda = \Omega(\log k)$ with high probability. In this event, by using the flow-cut gap for multicommodity flows [34], we can conclude that the graph $\mathcal{B}$ can support a multicommodity flow on the original demands. Finally, since the hallucinated demands can be routed on G_H with congestion $O(\log^2 m)$ (with high probability), the edges in $G_F = G_S \cup G_H$ can support a multicommodity flow of all requests with congestion $O(\log^2 m)$. ∎

Now, Lemmas 6.4 and 6.3 bound the cost of the solution, and Lemma 6.5 establishes the congestion bounds. In particular, it follows that with probability $1 - k^{-3}$, the final solution $G_S(k) \cup G_H(k)$ has cost $O(\log m) \cdot \mathsf{Opt}$ and congestion $O(\log^2 m)$. This completes the proof of Theorem 1.3.

7. Conclusion. In this paper, we considered the energy-efficient routing problem with costs on edges. We obtained an $O(\log^\alpha k)$-ratio approximation algorithm and an $\tilde{O}(\log^{3\alpha+1} k)$-ratio online algorithm, where $\alpha > 1$ is the dynamic-power exponent. While a poly-logarithmic approximation ratio is necessary, an interesting open question is to obtain an approximation ratio of the form $g(\alpha) \cdot \log k$, where the poly-logarithmic factor is not exponential in α.

REFERENCES

[1] *Vision and roadmap: Routing telecom and data centers toward efficient energy use*, May 2009, http://www1.eere.energy.gov/manufacturing/datacenters/pdfs/vision_and_roadmap.pdf. Proceedings of Vision and Roadmap Workshop on Routing Telecom and Data Centers Toward Efficient Energy Use.

[2] A. Agrawal, P. N. Klein, and R. Ravi, *When trees collide: An approximation algorithm for the generalized steiner problem on networks*, SIAM J. Comput., 24 (1995), pp. 440–456.

[3] M. Andrews, S. Antonakopoulos, and L. Zhang, *Minimum-cost network design with (dis)economies of scale*, SIAM J. Comput., 45 (2016), pp. 49–66.

[4] M. Andrews, A. Fernández, L. Zhang, and W. Zhao, *Routing for energy minimization in the speed scaling model*, in INFOCOM, 2010, pp. 2435–2443.

[5] A. Antoniadis, S. Im, R. Krishnaswamy, B. Moseley, V. Nagarajan, K. Pruhs, and C. Stein, *Hallucination helps: Energy efficient virtual circuit routing*, in Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014, 2014, pp. 1141–1153.

[6] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts, *On-line routing of virtual circuits with applications to load balancing and machine scheduling*, Journal of the ACM, 44 (1997), pp. 486–504.

[7] Y. Aumann and Y. Rabani, *An o(log k) approximate min-cut max-flow theorem and approximation algorithm*, SIAM J. Comput., 27 (1998), pp. 291–301.

[8] B. Awerbuch and Y. Azar, *Buy-at-bulk network design*, in FOCS, 1997.

[9] B. Awerbuch, Y. Azar, and A. Fiat, *Packet routing via min-cost circuit routing*, in ISTCS, 1996.

[10] B. Awerbuch, Y. Azar, E. F. Grove, M.-Y. Kao, P. Krishnan, and J. S. Vitter, *Load balancing in the l_p norm*, in FOCS, 1995, pp. 383–391.

[11] N. Bansal, A. Gupta, R. Krishnaswamy, V. Nagarajan, K. Pruhs, and C. Stein, *Multicast routing for energy minimization using speed scaling*, in MedAlg, 2012, pp. 37–51.

[12] P. Berman and C. Coulston, *On-line algorithms for steiner tree problems (extended abstract)*, in STOC, 1997, pp. 344–353.

[13] D. Brooks, P. Bose, S. Schuster, H. M. Jacobson, P. Kudva, A. Buyuktosunoglu, J.-D. Wellman, V. V. Zyuban, M. Gupta, and P. W. Cook, *Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors*, IEEE Micro, 20 (2000), pp. 26–44.

[14] D. Chakrabarty, C. Chekuri, S. Khanna, and N. Korula, *Approximability of capacitated network design*, in IPCO, 2011, pp. 78–91.

964 [15] D. Chakrabarty, A. Ene, R. Krishnaswamy, and D. Panigrahi, *Online buy-at-bulk network*
965 *design*, 2015 IEEE 56th Annual Symposium on Foundations of Computer Science, (2015),
966 pp. 545–562.
967 [16] D. Chakrabarty, E. Grant, and J. Könemann, *On column-restricted and priority covering*
968 *integer programs*, in Integer Programming and Combinatorial Optimization (IPCO), 2010,
969 pp. 355–368.
970 [17] D. Chakrabarty, R. Krishnaswamy, S. Li, and S. Narayanan, *Capacitated network design*
971 *on undirected graphs*, in APPROX, 2013.
972 [18] M. Charikar, J. Naor, and B. Schieber, *Resource optimization in qos multicast routing of*
973 *real-time multimedia*, IEEE/ACM Transactions on Networking, 12 (2004), pp. 340–348.
974 [19] C. Chekuri, M. T. Hajiaghayi, G. Kortsarz, and M. R. Salavatipour, *Approximation algo-*
975 *rithms for nonuniform buy-at-bulk network design*, SIAM J. Comput., 39 (2010), pp. 1772–
976 1798.
977 [20] C. Chekuri, S. Khanna, and F. B. Shepherd, *Multicommodity flow, well-linked terminals,*
978 *and routing problems*, in STOC, 2005, pp. 183–192.
979 [21] G. Even, J. Naor, S. Rao, and B. Schieber, *Divide-and-conquer approximation algorithms*
980 *via spreading metrics*, J. ACM, 47 (2000), pp. 585–616.
981 [22] N. Garg, V. V. Vazirani, and M. Yannakakis, *Approximate max-flow min-(multi)cut theo-*
982 *rems and their applications*, SIAM J. Comput., 25 (1996), pp. 235–251.
983 [23] M. X. Goemans and D. P. Williamson, *A general approximation technique for constrained*
984 *forest problems*, SIAM J. Comput., 24 (1995), pp. 296–317.
985 [24] A. Gupta, R. Krishnaswamy, and K. Pruhs, *Online primal-dual for non-linear optimization*
986 *with applications to speed scaling*, CoRR, abs/1109.5931 (2011).
987 [25] A. Gupta, A. Kumar, M. Pál, and T. Roughgarden, *Approximation via cost sharing:*
988 *Simpler and better approximation algorithms for network design*, J. ACM, 54 (2007), p. 11.
989 [26] M. Hajiaghayi, R. Khandekar, G. Kortsarz, and Z. Nutov, *Capacitated network design*
990 *problems: hardness, approximation algorithms, and connections to group steiner tree*, in
991 Manuscript, 2013.
992 [27] M. Imase and B. M. Waxman, *Dynamic Steiner tree problem*, SIAM J. Discrete Math., 4
993 (1991), pp. 369–384.
994 [28] W. B. Johnson, G. Schechtman, and J. Zinn, *Best constants in moment inequalities for*
995 *linear combinations of independent and exchangeable random variables*, Ann. Probab.,
996 (1985), pp. 234–253.
997 [29] D. R. Karger, *Random sampling in cut, flow, and network design problems*, Mathematics of
998 Operations Research, 24 (1999), pp. 383–413.
999 [30] R. Khandekar, S. Rao, and U. V. Vazirani, *Graph partitioning using single commodity*
1000 *flows*, J. ACM, 56 (2009).
1001 [31] P. N. Klein, S. Rao, A. Agrawal, and R. Ravi, *An approximate max-flow min-cut relation*
1002 *for unidirected multicommodity flow, with applications*, Combinatorica, 15 (1995), pp. 187–
1003 202.
1004 [32] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, Addison-
1005 Wesley Publishing Company, USA, 2009.
1006 [33] F. T. Leighton and S. Rao, *Multicommodity max-flow min-cut theorems and their use in*
1007 *designing approximation algorithms*, J. ACM, 46 (1999), pp. 787–832.
1008 [34] N. Linial, E. London, and Y. Rabinovich, *The geometry of graphs and some of its algorith-*
1009 *mic applications*, Combinatorica, 15 (1995), pp. 215–245.
1010 [35] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.
1011 [36] S. A. Plotkin and É. Tardos, *Improved bounds on the max-flow min-cut ratio for multicom-*
1012 *modity flows*, Combinatorica, 15 (1995), pp. 425–434.
1013 [37] S. Rao and S. Zhou, *Edge disjoint paths in moderately connected graphs*, SIAM J. Comput.,
1014 39 (2010), pp. 1856–1887.
1015 [38] H. P. Rosenthal, *On the subspaces of L^p $(p > 2)$ spanned by sequences of independent random*
1016 *variables*, Israel J. Math., 8 (1970), pp. 273–303.
1017 [39] F. S. Salman, J. Cheriyan, R. Ravi, and S. Subramanian, *Approximating the single-sink*
1018 *link-installation problem in network design*, SIAM Journal on Optimization, 11 (2001),
1019 pp. 595–610.
1020 [40] P. D. Seymour, *Packing directed circuits fractionally*, Combinatorica, 15 (1995), pp. 281–288.
1021 [41] A. Wierman, L. L. H. Andrew, and A. Tang, *Power-aware speed scaling in processor sharing*
1022 *systems*, in INFOCOM, 2009, pp. 2007–2015.

1023 **Appendix A. Waterfilling Algorithm Analysis.**

1024 *Proof of Theorem 2.1.* Our proof is essentially a direct adaptation of Theorem 4.2

in [10] for the online load balancing problem, to the online routing setting. For better clarity in proof (and to closely mirror the proof of Theorem 4.2 [10]), we assume, for the remainder of the proof, that all the scaling-factors c_e of edges is 1. This is without loss of generality as we can assume each edge e with scaling factor c_e is sub-divided into c_e edges with unit scaling factor.

For the i^{th} request, let $P_i^{\mathcal{A}}$ denote the path that the waterfilling algorithm chooses to route the demand, and let P_i^* denote the flow path of the optimal solution. Similarly, let $\ell_e(i)$ denote the load of edge e after routing the i^{th} request, and let $\ell_e^*(i)$ denote the load on edge e by using the optimal routing for the first i requests. Finally, let the overall instance have k requests which arrive online, so the final load on any edge e is $\ell_e(k)$ using the algorithm's routing, and is $\ell_e^*(k)$ by using the optimal routing.

We use the following potential function $\Phi(i) = \sum_e \ell_e(i)^\alpha$ which tracks the total dynamic power after routing the first i requests. Now, by the greedy nature of our routing, note that

$$\begin{aligned}
\Phi(i+1) - \Phi(i) &\leq \sum_{e \in P_i^*} (\ell_e(i) + d_i)^\alpha - \ell_e(i)^\alpha \\
&\leq \sum_{e \in P_i^*} (\ell_e(k) + d_i)^\alpha - \ell_e(k)^\alpha \\
&\leq d_i \cdot \alpha \sum_{e \in P_i^*} (\ell_e(k) + d_i)^{\alpha-1} \\
&\leq \alpha \cdot d_i \sum_{e \in P_i^*} \left(c\ell_e(k)^{\alpha-1} + \left(d_i \left(\frac{\alpha-1}{\ln c} + 1 \right) \right)^{\alpha-1} \right), \text{ for any } c > 1.
\end{aligned}$$

Above, the first inequality follows from the greedy choice of our routing; the second and third inequalities use the convexity of the dynamic power function (recall $\alpha > 1$); and the fourth is from Lemma 4.1 of [10]. We can choose $c > 1$ to optimize our final competitive ratio. Now, summing over all i, we get

$$\Phi(k) \leq \alpha \cdot c \cdot \sum_e \ell_e^*(k) \ell_e(k)^{\alpha-1} + \alpha \left(\frac{\alpha-1}{\ln c} + 1 \right)^{\alpha-1} \sum_e \sum_{i \,:\, e \in P_i^*} d_i^\alpha$$

Next, we note that $\sum_e \sum_{i \,:\, e \in P_i^*} d_i^\alpha \leq \sum_e (\ell_e^*(k))^\alpha$ since $\alpha \geq 1$ and $\ell_e^*(k) = \sum_{i \,:\, e \in P_i^*} d_i$ by definition of $\ell_e^*(k)$. So by using this, and the fact that $\Phi(k) = \sum_e \ell_e(k)^\alpha$, we get

$$\sum_e \ell_e(k)^\alpha \leq \alpha \cdot c \cdot \sum_e \ell_e^*(k) \ell_e(k)^{\alpha-1} + \alpha \left(\frac{\alpha-1}{\ln c} + 1 \right)^{\alpha-1} \sum_e (\ell_e^*(k))^\alpha$$

We can now use Holder's inequality to get

$$\sum_e \ell_e(k)^\alpha \leq \alpha \cdot c \cdot \left(\sum_e (\ell_e^*(k))^\alpha \right)^{\frac{1}{\alpha}} \left(\sum_e \ell_e(k)^\alpha \right)^{\frac{\alpha-1}{\alpha}} + \alpha \left(\frac{\alpha-1}{\ln c} + 1 \right)^{\alpha-1} \sum_e (\ell_e^*(k))^\alpha$$

Now, as in the proof of Theorem 4.2 in [10], if $x^\alpha = (\sum_e \ell_e(k)^\alpha)/(\sum_e (\ell_e^*(k))^\alpha)$, then

$$x^\alpha \leq \alpha \cdot c \cdot x^{\alpha-1} + \alpha \left(\frac{\alpha-1}{\ln c} + 1 \right)^{\alpha-1}$$

It is now easy to see that we can bound x by $\Theta(\alpha^\alpha)$ by choosing c to be a large enough constant. For example, with $c = \mathsf{e}$ (base of natural logarithm), $x^\alpha \leq \mathsf{e}\alpha x^{\alpha-1} + \alpha^\alpha$, i.e. $x - \frac{\alpha^\alpha}{x^{\alpha-1}} \leq \mathsf{e}\alpha$, which implies $x \leq 2\mathsf{e}\alpha$. ∎

Appendix B. Probabilistic Inequalities.

THEOREM B.1 ([35]). *Let $X_1, X_2, ..., X_N$ be N independent random variables such that $\Pr[X_i = 0] = 1 - p_i$ and $\Pr[X_i = 1] = p_i$. Let $Y = \sum_{i=1}^{N} X_i$ and $\mu = \mathbb{E}Y$. Then for any $\delta > 0$, it follows that*

$$\Pr\left[Y \leq (1-\delta)\mu\right] \leq \exp(-\mu\delta^2/2).$$

THEOREM B.2 ([38, 28]). *Let $X_1, X_2, \ldots, X_N$ be independent non-negative random variables. Let $\alpha > 1$ and $K_\alpha = \Theta(\alpha/\log\alpha)$. Then it is the case that*

$$\left(\mathbb{E}[(\sum_i X_i)^\alpha]\right)^{1/\alpha} \leq K_\alpha \ \max\left(\sum_i \mathbb{E}[X_i], \left(\sum_i \mathbb{E}[X_i^\alpha]\right)^{1/\alpha}\right).$$

COROLLARY B.3 ([11]). *Let $p \geq 0$, and let $X_1, X_2, \ldots, X_n$ be independent random variables, each taking value D with probability $\min\{1, p\}$. Then $\mathbb{E}[(\sum_i X_i)^\alpha] \leq (K_\alpha)^\alpha \cdot (pN\,D^\alpha + (pND)^\alpha)$, where $K_\alpha = \Theta(\alpha/\log\alpha)$.*

Proof. For the case when $p \geq 1$, $X_i = D$ with probability 1, and hence we can conclude that $\mathbb{E}[(\sum_i X_i)^\alpha] = (ND)^\alpha$. For the case when $p \in [0, 1]$, $\mathbb{E}[X_i] = pD$, and $\mathbb{E}[X_i^\alpha] = pD^\alpha$. From this we can conclude that the upper bound in Theorem B.2 is $K_\alpha \max(pND, (pN)^{1/\alpha} D)$. Taking α^{th} powers and replacing the max by a sum, we get $\mathbb{E}[(\sum_i X_i)^\alpha] \leq (K_\alpha\, D)^\alpha((pN)^\alpha + pN)$.