Online and Stochastic Survivable Network Design*

Anupam Gupta† Ravishankar Krishnaswamy† R. Ravi‡

Abstract

Consider the *edge-connectivity survivable network design* problem: given a graph $G = (V, E)$ with edge-costs, and edge-connectivity requirements $r_{ij} \in \mathbb{Z}_{\geq 0}$ for every pair of vertices $i, j \in V$, find an (approximately) minimum-cost network that provides the required connectivity. While this problem is known to admit good approximation algorithms in the offline case, no algorithms were known for this problem in the *online* setting. In this paper, we give a randomized $\tilde{O}(r_{\max} \log^3 n)$-competitive online algorithm for this edge-connectivity network design problem that runs in time $O(m^{r_{\max}})$, where $r_{\max} = \max_{ij} r_{ij}$. Our algorithms use the standard embeddings of graphs into random subtrees (i.e., into *singly connected* subgraphs) as an intermediate step to get algorithms for higher connectivity. As a consequence of using these random embeddings, our algorithms are competitive only against oblivious adversaries.

Our results for the online problem give us approximation algorithms that admit *strict* cost-shares with the same strictness value. This, in turn, implies approximation algorithms for (a) the *rent-or-buy* version and (b) the (two-stage) *stochastic version* of the edge-connected network design problem with independent arrivals. If we are in the case when the underlying graph is complete and the edge-costs are metric (i.e., satisfy the triangle inequality), we improve on our results to give an $O(\log n)$-competitive deterministic online algorithm for the rooted version of the problem, and constant-factor approximation algorithms for the rent-or-buy and stochastic variants of SNDP.

1 Introduction

We consider the edge-connectivity version of the *survivable network design* problem (SNDP): given a graph $G = (V, E)$ with non-negative edge-costs $c(e)$, and edge-connectivity requirements $r_{ij} \in \mathbb{Z}_{\geq 0}$ for every pair of vertices $i, j \in V$, the goal is to find a subgraph $H = (V, E')$ with minimum cost $\sum_{e \in E'} c(e)$ such that H contains r_{ij} edge-disjoint paths between i and j. The problem is of much interest in the network design community, since it seeks to build graphs which are resilient to edge failures. Since the problem is NP-hard (it contains the Steiner tree problem as a special case), it has been widely studied from the viewpoint of approximation algorithms (refer to [GK11] for a survey of results). In fact, these connectivity problems were one of the earliest applications of the primal-dual method in this area which led, over a sequence of papers, to the development of an $O(\log r_{\max})$-approximation algorithm [GGP+94]. Subsequently, one of the first uses of iterative rounding in approximation algorithms led to a 2-approximation for this problem (and for the general problem of network design with weakly-supermodular functions) [Jai01].

In this paper we extend the study of survivable network design problems in two different directions. First, we study these problems in the *online* setting: we are given a graph with edge-costs, and an upper bound $r_{\max}$ on the connectivity demand[1]. A sequence of vertex pairs $\{i, j\} \in V \times V$ is presented to us over time, each

*A preliminary version of this paper appeared in STOC 2009.

†Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA. Supported in part by NSF awards CCF-0448095 and CCF-0729022, and an Alfred P. Sloan Fellowship. Email: anupamg,ravishan@cs.cmu.edu.

‡Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA 15213, USA. Supported in part by NSF grant CCF-0728841. Email: ravi@cmu.edu.

[1]We can remove this assumption of knowing $r_{\max}$ upfront by losing another $\log r_{\max}$ in our approximation guarantee by using a standard doubling argument used in online algorithms

with some edge-connectivity demand r_{ij}—at this point we may need to buy some edges to ensure that all the edges bought by the algorithm provide an edge-connectivity of r_{ij} between vertices i and j. The goal is to remain competitive with the optimal offline solution of the current demand set. To the best of our knowledge, no online algorithms were previously known for this problem even for the online rooted 2-connectivity problem (i.e., for the case where all the vertex pairs share a root vertex r and the connectivity requirement is 2 for all pairs)—in fact, in Appendix A, we show a lower bound of $\Omega(\min\{|\mathcal{D}|, \log n\})$ on the competitive ratio for this special case, where $\mathcal{D}$ is the set of terminal pairs given to the algorithm. This is in contrast to the case of online 1-connectivity (i.e., online Steiner forest) where the best online algorithm is $\Theta(\log |\mathcal{D}|)$-competitive [BC97].

Theorem 1.1 *For the edge-connected survivable network design problem, there is an $\alpha = O(r_{\max} \log^3 n)$-competitive randomized online algorithm against oblivious adversaries.*

A somewhat surprising ingredient of our proof is that we use distance-preserving embeddings into random trees (i.e., into singly connected structures) to get algorithms for higher connectivity. In our work, these embeddings allow us to simplify the cost structure of the network and to abstract out a latent set cover-type problem, where the cuts are the sets and we want to cover them using edges. We should point out that the use of such randomized embeddings also implies that our algorithm is competitive only against oblivious adversaries, who are not aware of the algorithm's random coin tosses. We also note that while computational aspects are often brushed aside in online analysis, our online algorithm can be implemented in time $O(m^{r_{\max}})$, and hence runs in polynomial time only for constant $r_{\max}$. Refer to Section 2 for a high-level overview of our techniques.

Stochastic and Rent-or-Buy Problems. Another direction to extend the edge-connectivity problem is the stochastic case when the instance is drawn according to a probability distribution. In this paper we consider the case when we have a product distribution: for each pair i, j of vertices we are given a probability p_{ij}, and are guaranteed that *tomorrow* each pair will flip their coins independently, and if the coin turns up heads, they would demand k-connectivity. (For simplicity we assume that all pairs have the the same connectivity requirement of k.) We can buy some edges today at cost $c(\cdot)$, but if we wait for the actual set $\mathcal{D}$, the edges will cost $\lambda c(\cdot)$ tomorrow, for a pre-specified inflation parameter $\lambda \geq 1$; the goal is to minimize the sum of the cost of edges bought today and the expected cost of augmentation edges bought tomorrow (at the inflated price). Given an α-approximation algorithm for the basic k-edge-connectivity problem that admits *β-strict cost-shares* (see Section 1.2 for a definition of cost-sharing schemes), it is known that one can get a randomized algorithm which is an $(\alpha + \beta)$-approximation for the stochastic version [GPRS04].

A similar result holds for the rent-or-buy version of k-connectivity, where we are given a set of $\{s_i, t_i\}$ pairs, and the goal is to define k-edge-disjoint paths between each s_i-t_i. If n_e different pairs use an edge e, the cost of edge e is defined to be $c(e) \times \min\{n_e, M\}$ for some threshold M, capturing the fact that there is some incremental cost for different pairs using the same edge, but at some point this cost tapers off (and we "buy" the edge). Again, an α-approximation for k-edge-connectivity with β-strict cost-shares gives an $(\alpha + \beta)$-approximation [GKPR03].

Theorem 1.2 *For constant k, the online algorithm for k-edge-connectivity is a polynomial-time α-approximation algorithm with α-strict cost-shares. Hence, the problems of rent-or-buy and two-stage stochastic k-edge-connectivity with independent decisions admit $2\alpha = O(k \log^3 n)$-approximations.*

The only previous results known for these versions of higher-connectivity problems were $O(1)$-strict cost-shares implicitly given by Chuzhoy and Khanna [CK08b], and independently (but explicitly) by Chekuri et al. [CGK+08] for the special case of *rooted* connectivity, where all pairs seek k-connectivity to a single source r (and hence to each other).

Metric Costs. Finally, we improve on these results for the special case when the underlying graph is complete and the costs satisfy the triangle inequality. Firstly, we extend the $O(\log n)$-competitive algorithm for online Steiner tree to get an $O(\log n)$-competitive online algorithm for the rooted version of the k-EC-ND problem where all terminals require k-edge-connectivity to a pre-specified root vertex. Then, for the stochastic and rent-or-buy versions of the survivable network design problem, we obtain constant factor approximations by designing $O(1)$-approximation algorithms that admit $O(1)$-strict *cost-shares*. At a high level, our ideas are derived from the strict cost-shares for Steiner forest [FKLS06], and the metric k-connectivity algorithms of Cheriyan and Vetta [CV05].

1.1 Related Work

Steiner network problems have received considerable attention in approximation algorithms: Agrawal et al. [AKR95] and Goemans and Williamson [GW95] used primal-dual methods to design approximation algorithms for Steiner forests and other 1-connectivity problems (and some higher connectivity problems where multiple copies of edges could be used). Klein and Ravi [KR93] gave an algorithm for the 2-connectivity problem, which was extended by Williamson et al. [WGMV95] and Goemans et al. [GGP$^+$94] to higher connectivity problems, yielding $O(\log k)$-approximation algorithms for k-connectivity, all using primal-dual methods. Jain [Jai01] gives an iterative rounding technique to obtain a 2-approximation algorithm for the most general problem of SNDP. These techniques have recently been employed to obtain tight results (assuming P $\neq$ NP) for network design with degree constraints [LNSS07, LS08, BKN08]. Vertex connectivity problems are less well-understood: [CVV03, KN03, FL08] consider problems of *spanning* k-connectivity, and provide approximation algorithms with varying guarantees depending on k. Fleischer et al. [FJW06] give a 2-approximation for vertex connectivity when all $r_{ij} \in \{0, 1, 2\}$. Recently, improved approximation algorithms have been given for the problem of *single source* k-vertex connectivity [CCK08, CK08a], culminating in a simple greedy $O(k \log n)$ algorithm [CK08b]. In fact, the papers [CK08a, CK08b] also implicitly give $O(k)$-strict cost-shares for the single-source vertex-connectivity problem. As far as we can see, their techniques do not apply in the case of general survivable network design where vertex pairs do not share a common root, nor do they imply online algorithms with adversarial inputs. Very recently, Chuzhoy and Khanna [CK09] gave a very simple $O(k^3 \log n)$-approximation algorithm for the k-vertex-connectivity network design problem via an elegant randomized reduction to a collection of *element-connectivity* network design problems. Again, we do not see how these can be made online, or made to admit good cost-sharing schemes. When the edges have metric costs, there are, quite expectedly, better approximation algorithms for vertex connectivity. Khuller and Raghavachari [KR96] gave $O(1)$-approximations for k-vertex-connected spanning subgraphs. Cheriyan and Vetta [CV05] later gave $O(1)$-approximations for the single source k-connected problem and a $O(\log r_{max})$-approximation for metric vertex-connected SNDP. Recently, Chan et al. [CFLY08] give constant factor approximations for several degree bounded problems on metric graphs. As for the inapproximability, Kortsarz et al. [KKL04] give $2^{\log^{1-\varepsilon} n}$ hardness results for the vertex-connected survivable network design problem.

Imase and Waxman [IW91] first considered the online Steiner tree problem and gave a tight $\Theta(\log |\mathcal{D}|)$-competitive algorithm. Awerbuch, Azar and Bartal [AAB04] generalized these results for the online Steiner forest problem, and subsequently Berman and Coulston [BC97] gave the same $\Theta(\log |\mathcal{D}|)$ guarantee. However, we do not see how to use these ideas for the general problem with higher connectivity. In this paper, we use the results of Alon et al. [AAA$^+$03] for the online (weighted) set cover problem; the ideas used in this paper have been extended by Alon et al. [AAA$^+$04] and Buchbinder and Naor [BN06] to get online primal-dual based algorithms for *fractional* generalized network design. We note that while we can solve the fractional version of the online k-connectivity problems using these techniques, we do not know how to round this fractional solution online.

The use of strict cost-shares to get algorithms for rent-or-buy network design appears in [GKPR07]. Approx-

imation algorithms for two-stage stochastic problems were studied in [IKMM04, RS04], and some general techniques were given by [GPRS04, SS06]; in particular, using strict cost-shares to obtain approximation algorithms for stochastic optimization problems appears in [GPRS04].

1.2 Preliminaries

1.2.1 The k-EC-ND Problem

For most of the paper, we will present our results in the form of the *k-edge-connected network design problem* (k-EC-ND), which is survivable network design where $r_{ij} \in \{0, k\}$—this is just for simplicity; our results extend to the more general survivable network design problem whilst incurring small additional losses in our competitiveness and approximation guarantees.

1.2.2 Notation

Consider the k-EC-ND problem, and let $\mathcal{D} \subseteq \binom{V}{2}$ denote the set of demand pairs that require k-connectivity. For the rest of the paper, we shall refer to demand pairs by using curly brackets, e.g., $\{s, t\}$, and use the regular brackets to denote edges following standard convention, like (u, v).

1.2.3 Strict Cost-Sharing Schemes

An α-approximation algorithm Alg is said to be β-strict for the k-EC-ND problem if, for each $\{s_i, t_i\} \in \mathcal{D}$, there exist cost-shares $\xi(\{s_i, t_i\})$ such that the following properties hold:

- $\sum_{\{s_i,t_i\}\in\mathcal{D}} \xi(\{s_i, t_i\}) \leq c(\mathsf{OPT})$, where OPT is an optimal solution which k-edge-connects all terminal pairs in $\mathcal{D}$.
- There is an efficient augmenting procedure Augment (which takes as input a terminal pair and outputs a set of edges) such that s_i and t_i are k-edge-connected in $\mathsf{Augment}(\{s_i, t_i\}) \cup \mathsf{Alg}(\mathcal{D} \setminus \{s_i, t_i\})$.
- For each $\{s_i, t_i\} \in \mathcal{D}$, the total cost of edges output by $\mathsf{Augment}(\{s_i, t_i\})$ is at most $\beta\, \xi(\{s_i, t_i\})$.

As mentioned in the introduction, it is known that if we have an α-approximation algorithm for a problem that admits β-strict cost-shares, we can then get randomized $(\alpha + \beta)$-approximation algorithms for the two-stage stochastic and rent-or-buy versions of the problem [GPRS04, GKPR03].

1.2.4 Cost-Shares from Online Algorithms

Given an α-competitive online algorithm Alg for k-EC-ND, order all possible vertex pairs in some universal canonical ordering, and feed the actual demands $\mathcal{D}$ in the induced ordering to Alg. Then for any $\{s_i, t_i\} \in \mathcal{D}$, define the cost-share $\xi(\{s_i, t_i\})$ to be $\frac{1}{\alpha}$ times the increase in total cost incurred by the online algorithm. By the α-competitiveness of the online algorithm, we have $\sum_i \xi(\{s_i, t_i\}) \leq \frac{1}{\alpha} \cdot \alpha\mathsf{OPT} = \mathsf{OPT}$. Moreover, the fixed ordering of the demands means that the augmentation cost for a demand pair s_i-t_i to $\mathsf{Alg}(\mathcal{D} \setminus \{s_i, t_i\})$ is at most the online algorithm's cost increase when we had presented s_i-t_i to it, i.e., $\alpha \cdot \xi(\{s_i, t_i\})$.

2 The Basic Idea, at a High Level

Imagine we want to convert the connectivity augmentation problem into a hitting set problem: we are given a subgraph H of G that has l-edge-connected a demand pair s_i-t_i (where $l < k$), and we want to $(l + 1)$-edge-connect them. If we think of the s_i-t_i cuts as sets, then we would like to "hit" all these s_i-t_i cuts with edges. This is clearly doomed, since there are $M = 2^{n-1}$ cuts, and an $O(\log M)$-approximation for hitting set will be useless.

We could do better by noting that each minimal s_i-t_i cut in H is given by only l edges. While this bounds the number of cuts in H by $M = \binom{m}{l}$, the subgraph H might contain only a small fraction of G, and there may be many more cuts in G corresponding to the same cut in H—even an exponential number, and we are back to square one. Alternately, we could try to overcome this by hitting the cuts by *paths* connecting two nodes in H (instead of hitting the cuts by edges in G), but there could be exponentially many such paths, and this seems like another bad idea.

What the results in Section 3 show is that this is *not* a bad idea at all if we are slightly careful. Loosely speaking, if we take a random distance-preserving spanning subtree $T \subseteq G$, then we show that we can augment the connectivity using only the fundamental cycles (the cycle formed by any non-tree edge (u, v) along with the tree path between u and v) with respect to this spanning tree T. Interestingly, the (random) distance-preserving property allows us to control the cost of these connectivity augmentations. Furthermore, there are only at most m such fundamental cycles, and this enables us to get a compact hitting set instance. Of course, this high-level view oversimplifies things a bit: read on for the complete details. In Sections 3.1-3.3 we show how we can hit cuts by a small number of cycles/paths, and then Sections 3.4 and 3.5 use these ideas to develop our algorithms.

3 Online k-EC-ND on General Graphs

In this section, we present a $\widetilde{O}(k \log^2 m \log n)$-competitive online algorithm for the k-EC-ND problem on general graphs with demand set $\mathcal{D} \subseteq \binom{V}{2}$. We show how this also gives us $\widetilde{O}(k \log^2 m \log n)$-strict cost-shares, and hence implies polylogarithmic approximation (for constant values of k) guarantees for the rent-or-buy and stochastic k-EC-ND problems (see Section 1.2).

3.1 Embedding into Backboned Graphs

One of the major advantages of network design problems which only sought 1-edge-connectivity is that one can embed the underlying metric space into random trees [Bar96, FRT04, EEST05, ABN08], where the problems are easier to (approximately) solve. Such a reduction seems impossible even for 2-edge-connectivity as the problem is trivially infeasible on a tree. However, the simple but crucial observation is to not ignore these ideas, as we show below.

Given a graph $G = (V, E)$ with edge lengths/costs $c(e)$, probabilistically embed it into a *spanning* subtree (which we call the *base tree*) using the results of Elkin et al. and Abraham et al. [EEST05, ABN08]. Formally, this gives a random spanning tree $T = (V, E_T \subseteq E)$ of G with edge lengths $\widehat{c}_T$, such that for all $x, y \in V$:

- $\widehat{c}_T(e) = c(e)$ for all edges $e \in E_T$, and hence $d_T(x, y) \geq d_G(x, y)$; and
- $\mathbb{E}[d_T(x, y)] \leq \widetilde{O}(\log n) \cdot d_G(x, y)$, where d_G is the graph metric according to the edge lengths $c(e)$.

The distance d_T is defined in the obvious way: if $P_T(u, v)$ is the unique u-v path in T, then $d_T(u, v) = \sum_{e \in P_T(u,v)} \widehat{c}_T(e)$.

Now *instead of throwing away non-tree edges,* imagine each non-tree edge $e = (u, v) \in E \setminus E_T$ being given a new weight $\widehat{c}_T(e) = \max\{c(e), d_T(u, v)\}$. This suggests the following definition.

Definition 3.1 *A graph $G = (V, E)$ with edge-costs $c : E \to \mathbb{R}$ is called a* backboned graph *if there exists a spanning tree $T = (V, E_T)$ with $E_T \subseteq E$ such that all edges $e = (u, v) \notin E_T$ have the property that $c(e) \geq d_T(u, v)$. In this case, T is called the* base tree *of G.*

Note that the embeddings of [EEST05, ABN08] probabilistically embed graphs into a distribution $\mathcal{T}$ over backboned graphs (indexed by their base trees) with small expected stretch, i.e., $\mathbb{E}_{T \sim \mathcal{T}}[\widehat{c}_T(e)] \leq \widetilde{O}(\log n) c(e)$ for all $e \in G$. This show that for any subgraph H, the expected cost $\mathbb{E}_{T \sim \mathcal{T}}[\widehat{c}_T(H)] \leq \widetilde{O}(\log n) c(H)$. Finally,

since $\widehat{c}_T(e) \geq c(e)$ for all $e \in E, T \in \mathcal{T}$, we can bound the cost of any subgraph H' w.r.t edge costs c by the corresponding cost $\widehat{c}_T(H')$. We then get the following theorem.

Theorem 3.2 *A β-competitive online algorithm for* k-EC-ND *on backboned graphs implies a randomized $\beta \times \widetilde{O}(\log n)$-competitive algorithm for* k-EC-ND *on general graphs (against oblivious adversaries). Also, β-approximation algorithms for* k-EC-ND *on backboned graphs imply randomized $\beta \times \widetilde{O}(\log n)$ approximation algorithms on general graphs.*

Hence, for the subsequent sections (except those for the metric instances) we will assume that the input graph is a backboned graph, and will use its properties to design online and "cost-sharing" approximation algorithms.

3.2 Online 2-Edge-Connectivity

As a warm-up, consider the special case of k-EC-ND on a backboned graph $G = (V, E)$, when the connectivity requirement is $k = 2$ for all demand pairs, and furthermore, the problem instance is rooted, i.e., all demand pairs are of the form $\{r, t_i\}$ for some fixed root $r \in V$. A natural approach for this problem would be to first 1-edge-connect the root with the terminal which has arrived, and then augment connectivity in the next phase. Because the graph is backboned, it is easy to see that the optimal offline subgraph which just 1-edge-connects a set of terminals $T = \{t_1, t_2, \ldots, t_l\}$ with root r is the collection of base tree paths $\cup_{i=1}^{l} P_T(r, t_l)$. Therefore, the online 1-connectivity problem becomes trivial on backboned graphs—when a new terminal t_i arrives, we simply buy the base tree path $P_T(r, t_i)$, and this is optimal.

We now see how we can augment edges to 2-connect the terminals with the root, in an online fashion. Consider the stage in the online algorithm when a terminal t_i has arrived. As a first step, like mentioned above, we 1-edge-connect t_i to the root r by buying the path $P_T(r, t_i)$. Now if t_i is not 2-edge-connected to r in the current subgraph, then there must exist a cut-edge $e = (x, y)$ on the path $P_T(r, t_i)$. Removing the edge e also cuts the base tree T into 2 components—call the one containing the root as C_r, and the other containing the terminal t_i as C_{t_i}. Since e is the only tree-edge crossing this cut, there must exist a non-tree edge $f = (u, v)$ in OPT (an optimal offline solution which 2-edge-connects the current set of terminals with r) such that f crosses the cut (C_r, C_{t_i}) (and as a consequence, observe that the edge e would be contained in the base tree path $P_T(u, v)$).

The crucial observation now is the following: if we were to include the *entire cycle* $O_{(u,v)} = P_T(u, v) \cup (u, v)$ to our current subgraph, then e would no longer be a cut-edge separating r from t_i (because there is now an alternate path from x to y in $O_{(u,v)}$). Furthermore, we can use the backbone property of G and in fact charge the cost of the entire cycle to the single edge f that OPT bought.

At a high level, this motivates modeling the augmentation problem as the following online set cover instance, where **(i)** the elements are the tree-edges, **(ii)** the sets S_{uv} correspond to the cycles $O_{(u,v)}$, and **(iii)** an element/tree-edge e is "covered" by a set S_{uv} if and only if $e \in O_{(u,v)}$. By the preceding arguments, we can see that the cost of an optimal offline solution to cover *all* the cut-edges on the paths $P_T(r, t_i)$ is 2OPT. Hence, by the polylogarithmic competitiveness of the online set cover algorithm of Alon et al. [AAA+03], we would get an online 2-edge-connectivity algorithm with polylogarithmic guarantees. In what follows, we formalize this intuition, and generalize it to the setting of k-EC-ND.

3.3 A Small Collection of Covering Cycles

In this section, we show how we can augment connectivity (from, say, l to $l + 1$) for a demand pair $\{s_i, t_i\}$ by showing that all its minimal cuts can be *covered* by a small collection of fundamental cycles (w.r.t the base tree T) of low cost. For the case when $l = 1$, this is just the set cover instance outlined in the previous section. Before we state our Cut Cover Theorem, we begin with some notation that will be useful for the rest of this section.

Notation: *Base Cycles.* Let G be a backboned graph that is an instance of the k-EC-ND problem with demand set $\mathcal{D}$, and let T be the base tree in G. For any edge $e = (u, v) \notin E_T$, define the *base cycle* O_e to be the fundamental cycle $\{e\} \cup P_T(u, v)$ of e with respect to T.

Now, let H be a subgraph which l-edge-connects (for some $l < k$) the vertices s_i and t_i for some demand pair $\{s_i, t_i\} \in \mathcal{D}$, and suppose H also contains the base tree path $P_T(s_i, t_i)$. The l-edge-connectivity assumption implies there are l edge-disjoint paths from s_i to t_i in H: denote this set of edge-disjoint paths by $\mathcal{P}_i$. Clearly, any l-cut (a set of l edges removing which would separate s_i and t_i in H) in H must pick exactly one edge from each path in $\mathcal{P}_i$: we define $\mathsf{viol}_H(i)$ to be the set of all such l-cuts.

Labeling: Consider any cut $Q \in \mathsf{viol}_H(i)$. Since Q is a minimal l-cut for the demand pair s_i-t_i in H, it must be that any end vertex of a cut edge is reachable from one of s_i or t_i in $H \setminus Q$. We label each end vertex v reachable from s_i in $H \setminus Q$ by L (i.e., we set $\mathsf{label}(v) = L$), and each end vertex v reachable from t_i by R (we set $\mathsf{label}(v) = R$). Every other vertex in $V(G)$ has a label U; hence all but at most $2|Q|$ nodes are labeled U, which we denote by a "trivial label". Note that the labeling of the end vertices of a cut Q depends on the subgraph H and not just the set of edges in Q.

Theorem 3.3 (Cut Cover Theorem) *Consider a* k-EC-ND *instance $\mathcal{I}$, and let* OPT *denote any optimal solution. Let $H \subseteq G$ be any subgraph that l-edge-connects terminal pair $\{s_i, t_i\}$ for some $l < k$, such that the base tree path $P_T(s_i, t_i) \subseteq H$. Then for any l-cut $Q \in \mathsf{viol}_H(i)$, given the labeling of the endpoints of Q as described above, we can find an edge $e = (u, v) \in E(\mathsf{OPT})$ such that $O_e \setminus Q$ connects some L-vertex to some R-vertex. This ensures that s_i and t_i are connected in $(H \cup O_e) \setminus Q$.*

Note that the algorithms in Sections 3.4 and 3.5 depend only on the statement of the above Cut Cover Theorem 3.3, so readers strapped for time can jump straight to the algorithms.

Proof Outline. The proof is by contradiction and assumes that there is no edge e s.t the base cycle O_e can cover this cut. We first give the outline of the proof, which would help in following the sequence of Lemmas 3.4–3.7. Suppose we delete all the edges of some minimal cut $Q \in \mathsf{viol}_H(i)$ from the current subgraph H. Such a cut (in particular, removing the edges $Q \cap T$) will separate the base tree into $|Q \cap T| + 1$ components (denoted by $\mathcal{C}$), with s_i and t_i belonging to different components (see Figure 3.1 for an illustration where the different circles are the tree components).

Firstly, observe that every component $C \in \mathcal{C}$ will have at least one vertex with a non-trivial label (a vertex with a label not U) since it has at least one edge from $Q \cap T$ in its boundary. To get the main intuition behind the proof, let us make the simplifying assumption that each component contains vertices of only one non-trivial label. If a component has its only non-trivial labels as L-vertices, we refer to it as an L-component, and likewise R-components only contain R-vertices.

Now, since OPT can $(l + 1)$-edge-connect s_i and t_i, it means that there must exist a path P^* including which would connect these 2 vertices in H. We focus on this path and traverse it edge by edge, starting from s_i. Suppose we are considering the j^{th} edge on this path, and suppose it begins from an L-component. Then, in Lemma 3.4, we show that this edge must also end in an L-component—otherwise, its base cycle would cover this cut Q. This then lets us inductively proceed and show that each edge will always terminate in an L-component, since we begin from s_i and it belongs to an L-component. Hence we would never reach t_i (contained in an R-component), which gives us the desired contradiction.

In general, it may not be the case that a component has only L or R-vertices. To handle this, we extract out a subset of edges of the path P^* (called the Canonical Sequence), and argue that the tree edges in $Q \cap T$ which are induced by the base cycles w.r.t the canonical sequence, will satisfy this "consistency" property (Lemma 3.6). This is sufficient to push the induction through and we would arrive at the same contradiction. We now present the complete details.

Proof. Consider a cut $Q \in \mathsf{viol}_H(i)$. Note that $Q \cap T \neq \emptyset$, since by our assumption the base tree path $P_T(s_i, t_i) \subseteq H$ and hence the cut Q must contain some edge on it. Let the edges $Q \cap T$ separate the base tree into into $t \leq l + 1$ components $\mathcal{C} = \{C_1, C_2, \ldots, C_t\}$. The terminals s_i and t_i must belong to different components: let $C(s_i)$ and $C(t_i)$ denote the components containing them. In general, let $C(v)$ denote the component containing vertex v. A component $C \in \mathcal{C}$ is called a *star component* if it contains some vertex from $P_T(s_i, t_i)$. We refer to the edges in $Q \cap T$ as *portal edges*. For every component $C \neq C(t_i) \in \mathcal{C}$, let the *parent edge* $\mathsf{head}(C)$ be the first portal edge on the base tree path from any vertex in C to t_i; note that the component $C(t_i)$ does not have a parent edge. Also note that for non-star components, $\mathsf{head}(C)$ also happens to be the first portal edge on the base tree path from any vertex in C to s_i.

For example, in Figure 3.1, the dashed edges are the portal edges, $\mathsf{head}(C_2)$ is the portal edge between C_2 and C_1, $\mathsf{head}(C_6)$ is the portal edge between C_6 and C_5, and C_1, C_4, C_5, C_{10} are the star components.

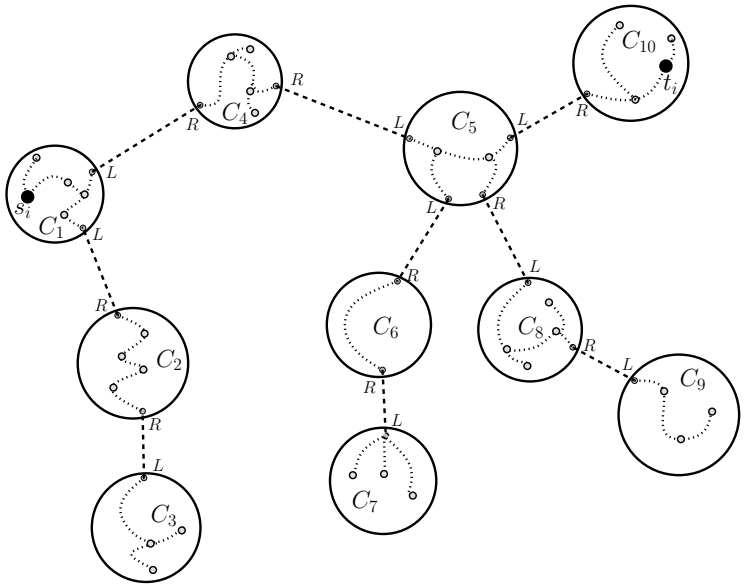


Figure 3.1: Example of the Portal Graph: circles are components, the dashed edges are portal edges, and dotted edges are other base-tree edges.

Since each edge in Q belongs to a distinct path in $\mathcal{P}_i$ (Q is a minimal l-cut separating s_i from t_i), the end vertices of any portal edge—and indeed of any edge in Q—have distinct labels from the set $\{L, R\}$. For a portal edge e, say its L-vertex is its unique endpoint labeled L, and its other endpoint is its R-vertex.

To prove Theorem 3.3, we will show that there exists an edge $e = (u, v) \notin Q$ which lies in an optimal solution such that $O_e \setminus Q$ contains a path between an L-vertex and an R-vertex in Q; in turn, this will ensure that s_i and t_i are connected in $(H \cup O_e) \setminus Q$, completing the proof. For the remainder of the proof, an edge (u, v) which satisfies this property is said to *cover* the cut Q.

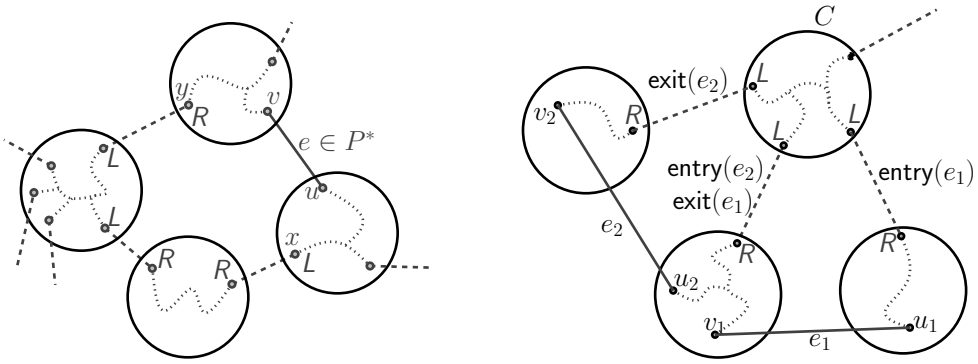
THE CANONICAL SEQUENCE: Since s_i and t_i can be k-edge-connected in G, there must be a s_i-t_i path P^* *contained in the optimal* k-EC *subgraph* with $P^* \cap Q = \emptyset$. We first eliminate some "redundant" edges from P^* and show that among the other edges, there is one that covers Q. First remove all edges from P^* that are internal to some component in $\mathcal{C}$. Now consider a new undirected graph—the *component graph*—whose vertex set is the collection $\mathcal{C}$ of components, and there is an edge (C_i, C_j) when there is an edge $(u, v) \in P^*$ such that $u \in C_i$ and $v \in C_j$. The edges in P^* now correspond to a path (not necessarily simple) between $C(s_i)$ and $C(t_i)$ in the component graph. We then remove edges from P^* that correspond to cycles in the component

graph, and are left with a set of edges P^* corresponding to a simple path between $C(s_i)$ and $C(t_i)$ in the component graph. Say the edges of P^* in this order are $\langle e_1 = (u_1, v_1), e_2 = (u_2, v_2), \ldots, e_p = (u_p, v_p)\rangle$. Note that $C(u_i) = C(v_{i-1})$ for $2 \leq i \leq p$; however u_i need not be the same as v_{i-1} in general. We refer to this resulting sequence of edges also as P^* and call it the *canonical* sequence, and all the components $C(u_j)$ the *canonical* components.

For a contradiction, suppose there is no edge $(u, v) \in P^*$ that covers the cut Q. We now prove a set of lemmas about the canonical sequence and the labeling of the portal edges to show that this cannot happen. Recall that each portal edge has different labels from $\{L, R\}$ on its endpoints. When tracing some u-v path in the base tree T, we say some portal edge is crossed with *signature* $(L \to R)$ if the endpoint labeled L is closer to u than to v in the base tree T. Clearly, the signature of the portal edge depends on the starting vertex u and ending vertex v of the path.

Lemma 3.4 (Alternating Paths Lemma) *Suppose none of the edges $(u', v') \in P^*$ covers Q. For any edge $(u, v) \in P^*$, if the first portal edge on $P_T(u, v)$ is crossed with signature $(L \to R)$, then the final portal edge on $P_T(u, v)$ is crossed with signature $(R \to L)$. Also, the portal edges crossed along the way have alternating signatures $(L \to R)$, $(R \to L)$, …, $(L \to R)$, $(R \to L)$. An analogous statement is true in the case the first portal edge is crossed with signature $(R \to L)$.*

Proof. If the u-v base tree path $P_T(u, v)$ first crosses a portal edge with signature $(L \to R)$ and also ends by crossing a portal edge with signature $(L \to R)$, the base cycle $O_{(u,v)}$ would connect the first L-vertex in $C(u)$ to the final R-vertex in $C(v)$. Moreover, the portions of $O_{(u,v)}$ within $C(u)$ and $C(v)$ are disjoint from Q, and hence $O_{(u,v)} \setminus Q$ would connect these L and R vertices, contradicting the fact that (u, v) does not cover Q. For example, in Figure 2(a) we see that x and y would be connected in $O_{(u,v)} \setminus Q$.



(a) Alternating Paths Lemma

(b) (u_1, v_1) and (u_2, v_2) transit C

Figure 3.2: Illustrative figures for the proof. Again, the dashed edges are portal edges, dotted edges are other base-tree edges, and solid edges belong to P^*.

Likewise, if the path $P_T(u, v)$ enters some component C through a portal edge signed $(L \to R)$ and also exits C through an $(L \to R)$ edge, the portion of $O_{(u,v)}$ within the component C would connect the entry vertex labeled R and exit vertex labeled L in $O_{(u,v)} \setminus Q$; as a result, including the edges of $O_{(u,v)}$ to H would connect s_i and t_i (even if the edges Q are deleted) by the definition of L and R-vertices. This is also a contradiction, completing the proof. ∎

Lemma 3.5 (Star-Path Lemma) *Suppose no $(u', v') \in P^*$ covers Q. Consider the portal edges $e'_1, e'_2, \ldots, e'_s$ when traversing the s_i-t_i path $P_T(s_i, t_i)$ on the base tree T. Then the signatures of these edges alternate $(L \to R)$, $(R \to L)$, …, $(L \to R)$.*

Proof. If we have two consecutive portal edges e'_j and e'_{j+1} that are signed $(L \to R)$, then we would have an L-vertex and an R-vertex, both of which lie on the path $P_T(s_i, t_i)$, belonging to the same (star) component C. Since we assume that H contains $P_T(s_i, t_i)$, these two vertices would be connected in $H \setminus Q$, thus contradicting the fact that Q is itself a violated cut separating s_i from t_i. ■

Lemma 3.6 (Consistency Lemma) *Suppose no $(u', v') \in P^*$ covers Q. Consider a component $C \neq C(t_i)$ such that* $\mathsf{head}(C)$*'s L-vertex belongs to C. Then, for any $(u, v) \in P^*$, if $P_T(u, v)$ intersects the component C, the portal edge $P_T(u, v)$ takes when entering C (if any) has its L-vertex in C. The same is true for the portal edge $P_T(u, v)$ takes when exiting C (if any). An analogous statement holds if* $\mathsf{head}(C)$*'s R-vertex is contained in C.*

Proof. Let $\mathsf{entry}(u, v)$ and $\mathsf{exit}(u, v)$ denote the portal edges used by $P_T(u, v)$ to enter and exit C respectively if we traverse $P_T(u, v)$ *from u to v*. For an edge $(x, y) \in P^*$, we say (x, y) *transits* a component C if $P_T(x, y)$ intersects C, but neither x nor y belong to C. (see Figure 2(b) for an example.)

We first consider the case when C is not a star component *and* is a canonical component; the proof for the C not being canonical is only simpler and we later consider C being a star component.

Let $\langle e'_1, e'_2, \ldots, e'_a \rangle \subseteq P^*$ be the edges in P^* which transit C (in that order) before some edge $e_1 \in P^*$ has an endpoint in C. (Such an edge e_1 exists because we have assumed that C is a canonical component.) The subsequent edge $e_2 \in P^*$ exits C, and let $\langle e''_1, e''_2, \ldots, e''_b \rangle \subseteq P^*$ be the following edges that transit C. Since C is not a star component, $\mathsf{entry}(e'_1)$ and $\mathsf{exit}(e''_b)$ must be the edge $\mathsf{head}(C)$, which by the assumption of the lemma has its L-vertex in C. This is because, if we shrink all the components and trace the path taken by P^* along the tree formed by just the portal edges in $Q \cap T$, the first time we visit C has to be via its head edge, by the definition of head edges. Likewise, the final edge to visit C must leave along the same head edge, since eventually this path ends up in t_i. Furthermore, by Lemma 3.4, $\mathsf{exit}(e'_1)$ must have its L-vertex in C as well.

Moreover, since the base path traversals are all done along the tree T, it is not hard to see that $\mathsf{entry}(e'_{j+1}) = \mathsf{exit}(e'_j)$ for $1 \leq j < a$. Inductively applying the alternating paths lemma, all the portal edges $\mathsf{entry}(e'_j)$ and $\mathsf{exit}(e'_j)$ have their L-vertices in C. Since $\mathsf{entry}(e_1) = \mathsf{exit}(e'_a)$, we also get that $\mathsf{entry}(e_1)$ has its L-vertex in C. The same inductive argument applied starting with e''_b and working backwards shows that the entry and exit edges used by e''_j for all j, and e_2 all have their L-vertices in C. For the case when C is not a canonical component, the argument is only simpler, since we would not have the edges e_1 and e_2 and have only a set of transiting edges.

Finally, when C is a star component, it is no longer true that the edge $\mathsf{entry}(e'_1)$ is the same as $\mathsf{head}(C)$. However, either $C = C(s_i)$ (in which case the proof is the same as above without any edges e'_j or e_1), or else $\mathsf{entry}(e'_1)$ must be the head edge for the previous star component C_{prev} on the s_i-t_i path. Hence, since $\mathsf{head}(C)$ has its L-vertex in C, the Star-Path Lemma 3.5 implies that $\mathsf{entry}(e'_1) = \mathsf{head}(C_{prev})$ also has its L-vertex in C. Now the rest of the proof is identical to that above. ■

Lemma 3.7 (Final Component Lemma) *Suppose there is no $(u', v') \in P^*$ that covers Q. For any $(u, v) \in P^*$ such that $P_T(u, v)$ intersects $C(t_i)$, the portal edge taken to enter $C(t_i)$ has its R-vertex in C_{t_i}. The same is the case for the portal edge taken to exit $C(t_i)$, if any.*

Proof. The proof of the lemma is very similar to that for Lemma 3.6. Since C_{t_i} is the final component on the path P^*, we would not have the edges of the form e_2 and e''_j (there is only one edge in P^* that has an end vertex in C_{t_i}, since we have eliminated all cycles). Also, because e'_1 is the first edge in P^* to transit $C(t_i)$, $\mathsf{entry}(e'_1)$ must be the head edge for the previous star component C_{prev} on the s_i-t_i path. From the Star-Path Lemma 3.5, we get that $\mathsf{entry}(e'_1)$ has its R-vertex in $C(t_i)$. By Lemma 3.4, $\mathsf{exit}(e'_1)$ must have its R-vertex in $C(t_i)$ as well. Like in the previous proof, $\mathsf{entry}(e'_{j+1}) = \mathsf{exit}(e'_j)$ for $1 \leq j < a$. Inductively applying the alternating

paths lemma, all the portal edges $\mathsf{entry}(e'_j)$ and $\mathsf{exit}(e'_j)$ have their R-vertices in C. Since $\mathsf{entry}(e_1) = \mathsf{exit}(e'_a)$, we also get that $\mathsf{entry}(e_1)$ has its R-vertex in C. ■

To complete the proof of Theorem 3.3, we argue the following.

Lemma 3.8 *Suppose no $(u', v') \in P^*$ covers Q. Then for all vertices v_j belonging to P^* (for $1 \leq j \leq p$), we have $C(v_j) \neq C(t_i)$.*

Proof. The proof is an induction on j, for $1 \leq j \leq p$. Since $C(u_1) = C(s_i)$, we know that $\mathsf{head}(C(u_1))$ has its L-vertex in C_{u_1}. By the Alternating Paths Lemma 3.6, we know that the final portal edge on $P_T(u_1, v_1)$ must have its L-vertex in $C(v_1)$. This implies that $C(v_1) \neq C(t_i)$, otherwise we would violate the Final Component Lemma 3.7. This establishes the base case. Now, since $C(v_1) \neq C(t_i)$, the Consistency Lemma 3.6 implies that $\mathsf{head}(C(v_1))$ must have its L-vertex in the component $C(v_1)$. But because $C(u_{j+1}) = C(v_j)$ for all j, we have that $\mathsf{head}(C(u_2))$ must have its L-vertex in $C(u_2)$, and therefore we can proceed inductively. ■

But note that Lemma 3.8 implies that we never reach $C(t_i)$ while following the canonical path, which contradicts the fact that P^* corresponds to a path between $C(s_i)$ and $C(t_i)$ in the component graph. This contradiction completes the proof, and hence implies that there must be some edge $(u, v) \in P^*$ that covers the cut Q. ■

3.4 Augmentation using Hitting Sets

We now show how we can use the covering property to get a low-cost augmentation. Given an instance $G, c(\cdot)$ of the k-EC-ND problem, suppose we have a subgraph H such that all terminal pairs $\{s_i, t_i\}$ are l-edge-connected in H: we now identify *sets* and *elements* such that the HITTING SET problem exactly captures the problem of augmenting H to $(l+1)$-edge-connect every s_i to t_i. Moreover, we want to do this in a way such that the number of sets and elements is small; if we were allowed exponentially many sets, we could imagine each l-cut $(U, V \setminus U)$ that separates s_i from t_i to be a set, and the edges of $G \setminus H$ to be the elements, such that element/edge e belongs to the set/cut $(U, V \setminus U)$ if $e \in \partial U$. But this gives us too many sets, as mentioned in Section 2.

To do this more efficiently, consider this: we can imagine H already contains the base tree, since it costs at most as much as the optimum k-EC solution. Now look at the following hitting set instance $\mathcal{I}_A$: for each violated l-cut $Q \in \mathsf{viol}_H(i)$ for a terminal pair $\{s_i, t_i\}$, we have a *set* in our instance. (Recall that now $Q \subseteq E$ can be just a set of edges.) In case the same set of edges Q separate several terminal pairs, we have a set for each terminal pair. For each edge $e = (u, v)$ in G we have an element. An element/edge e belongs to a set/cut Q if the edge e covers the cut Q—in other words, if the base cycle O_e satisfies the property that $(H \cup O_e) \setminus Q$ connects the terminal pair $\{s_i, t_i\}$. The cost of an element e is simply the cost of the base cycle O_e, which is at most $2c(e)$, by the properties of the backboned graph. Note that in this instance, the number of sets is at most $|\mathcal{D}| \cdot m^l = O(n^2 m^l)$ and the number of elements is at most m. A straightforward consequence of the Cut Cover Theorem 3.3 establishes the following:

Theorem 3.9 (Augmentation Theorem) *Given an backboned instance $G, c(\cdot)$ of the k-EC-ND problem, suppose we have a subgraph H containing the base tree such that the terminal pairs $\{s_i, t_i\}$ are l-edge-connected in H, for some $l < k$. Then the instance of the hitting set problem $\mathcal{I}_A$ created above has a solution costing at most $2\,c(\mathsf{OPT})$. Furthermore, if the set of elements/edges bought in a solution to the hitting set instance is F, then the subgraph $(H \cup (\cup_{e \in F} O_e))$ is a network that $(l+1)$-edge-connects every terminal pair $\{s_i, t_i\}$.*

As a warm-up, this shows that we can solve the k-EC-ND problem, and more generally the generalized Steiner connectivity problem, on any backboned graph by starting off with the base tree as the 1-edge-connected network, and repeatedly applying Theorem 3.9 (and a good approximation algorithm for hitting set) to augment

the connectivity from l to $l+1$ at cost $O(\log(n^2 m^l))c(\mathsf{OPT})$. In total, this approach gives us an approximation guarantee of $\sum_l O(l \log m + \log n) = O(r_{\max}^2 \log m + r_{\max} \log n)$. Finally, translating this to general networks loses another almost-logarithmic factor via Theorem 3.2. However, we can do better (and even do it online), as we now show.

3.5 Online Algorithm using Hitting Sets

To give an online algorithm for k-EC-ND, let us consider the above proofs again. When we defined the hitting set instance $\mathcal{I}$ corresponding to the $(l+1)$-augmentation problem, it appeared as if the notion of an element/edge e hitting a set/cut Q depended on the subgraph H. However, this is not the case: recall that the Cut Cover Theorem 3.3 showed that for any l-cut $Q \in \mathsf{viol}_H(i)$, there exists an edge $e = (u, v) \in \mathsf{OPT}$ such that $O_e \setminus Q$ connects an L-vertex to an R-vertex. In fact, if we were only given some set of edges $\widehat{Q}$ and some labels on its endpoints, and the theorem gives us an edge e, then this edge e is good for *all* subgraphs H such that (i) $P_T(s_i, t_i) \subseteq H$, (ii) $\widehat{Q}$ is an l-cut separating s_i and t_i in H, and (iii) the labels are indeed the labels we would get given H and $\widehat{Q}$. Moreover, for any cut Q, once we know the L and R labels of its end vertices, we can also identify whether an element (u, v) covers the cut Q. These are the properties we exploit in our online algorithm.

For the online algorithm for backboned graph, we first set up an instance $\mathcal{I}$ of the HITTING SET problem:

- **Universe.** For each edge $e \in E$, we have an element; there are $N = m$ elements. The cost of element e is $c(O_e) \in [c(e), 2c(e)]$.
- **Sets.** For each $l \in \{1, 2, \ldots, (k-1)\}$, we have a collection $\mathcal{F}_l$ of $M_l \leq \binom{m}{l} 2^l$ sets, where each set Q^l is a set of l edges *along* with $\{L, R\}$ labels on the endpoints of these edges. Hence $\mathcal{F} = \cup_l \mathcal{F}_l$ are all the $M := O((2m)^k)$ sets.
- **Incidence.** A element $e = (u, v)$ hits a set Q^l if and only if the subgraph $O_{(u,v)} \setminus Q^l$ connects an L-vertex and an R-vertex in Q^l.

Now when a terminal pair $\{s_i, t_i\}$ arrives, we first buy the edges on s_i-t_i base tree path $P_T(s_i, t_i)$ that have not yet been bought, and then perform a series of $(k-1)$ augmentations. In round l, we feed all the minimal violated cuts with l edges in the current subgraph H along with their $\{L, R\}$ labels to the online hitting set algorithm, which results in a new subgraph with increased connectivity. Note that the deterministic online algorithm for weighted set cover given by Alon et al. [AAA+03] would be $O(\log N \log M) = O(k \log^2 m)$-competitive on this hitting set instance as well. Formally, the algorithm is presented below.

Algorithm 1 OnlineAlg($\mathcal{D}$) for online k-EC-ND on backboned graphs

let $H \leftarrow \emptyset$.
set up the instance $\mathcal{I}$ of online hitting set.
for each terminal pair $\{s_i, t_i\}$ that arrives **do**
  let $H \leftarrow H \cup P_T(s_i, t_i)$
  for $l = 1$ to $k - 1$ **do**
    while $\{s_i, t_i\}$ not $l + 1$-edge-connected in H **do**
      find some violated l-cut Q between s_i and t_i in H and its labeling w.r.t. H
      feed (Q, labeling) to online hitting set algorithm; let its output be $F \subseteq E$
      let $H \leftarrow H \cup (\cup_{e \in F} O_e)$
    end while
  end for
end for

Theorem 3.10 *The algorithm* OnlineAlg *is has a competitive ratio of* $O(k \log^2 m)$ *for the* k-EC-ND *problem on backboned graphs.*

Proof. The proof essentially reiterates the aforementioned facts. Consider the case where τ terminals have arrived, and let OPT be an optimal offline network k-edge-connecting the demand pairs $\{s_i, t_i\}_{i \le \tau}$. Clearly, the total cost spent in Step 4 in buying base tree paths is at most $c(\mathsf{OPT})$. Moreover, since for each request we feed the online algorithm, there is an element/edge $e \in \mathsf{OPT}$ that hits it (Theorem 3.3), the optimal offline cost to hit all our requests is at most $2c(\mathsf{OPT})$. (The factor 2 arises because we buy O_e with cost at most $2c(e)$, even though OPT may only buy e.) Hence, from the $O(\log M \log N)$-competitiveness of the online hitting set algorithm, we get $O(k \log^2 m)$-competitiveness for our online algorithm. ■

Combining this with Theorem 3.2, and with the discussion in Section 1.2, we immediately get:

Corollary 3.11 (Result for General Graphs) *There is an* $\widetilde{O}(k \log^2 m \log n)$*-competitive randomized online algorithm for the* k-EC-ND *problem on general graphs.*

Corollary 3.12 (Cost-Shares from Online Algorithms) *The (randomized)* α*-competitive* k-EC-ND *algorithm gives* α*-strict cost-shares for* k-EC-ND*. Hence, there is a randomized* 2α*-approximation for the rent-or-buy version of* k-EC-ND*, and also for the two-stage stochastic version with independent arrivals.*

4 The Complete Metric Case

In this section, we assume that G is a complete graph, and that the edge-costs $c(\cdot)$ satisfy the triangle inequality, i.e., $c(u, v) \le c(u, w) + c(w, v)$ for all $u, v, w \in V$. Under this assumption, we can improve on our results to give a deterministic $O(\log n)$-competitive online algorithm, and constant-factor approximation algorithms for the stochastic and rent-or-buy cases of k-EC-ND.

4.1 Online k-EC-ND on Metric Graphs

In this section, we consider the *rooted* version of the online k-EC-ND problem on complete metrics, and give a deterministic online $O(\log n)$-competitive algorithm. Formally, we are given a complete graph G with the costs on edges $c(\cdot)$ satisfying the triangle inequality, and a root vertex r; a new demand vertex v_i arrives on the i^{th} day requiring k-edge-connectivity to the root r. The goal is to buy a set of edges $\mathcal{E}_i$ on the i^{th} day such that the collection $\cup_{j=1}^{i} \mathcal{E}_j$ contains k-edge-disjoint paths from r to $v_{j'}$ for all $j' \in [1, i]$, and the cost $c(\cup_{j=1}^{i} \mathcal{E}_j)$ is competitive with the cost of an optimal offline subgraph which k-edge-connects the demand vertices $v_1, v_2, \ldots, v_i$ with root r.

At a high level, our idea for the online algorithm is the following: when a new terminal arrives, run the online algorithm for Steiner tree; for each edge (u, v) that it buys, we buy a *minimum cost* set of edges to k-edge-connect u and v. While such an algorithm would indeed return a feasible solution to the k-EC-ND instance, it may not always be $O(\log n)$-competitive. However, what we can show is that it would be $O(\log n)$-competitive provided the online Steiner tree constructed has *bounded degree* (i.e., every vertex has degree bounded by a constant).

We first describe our k-EC-ND algorithm assuming such a *good* online algorithm for the Steiner forest problem, and then show how we can modify the algorithm of Imase and Waxman [IW91] to get the properties we desire. In the following, let Alg denote the $O(\log n)$-competitive online algorithm for Steiner tree which always maintains a feasible solution of bounded degree, and let N_v represent the set of the k nearest neighbors around any vertex v in G.

Algorithm 2 OnlineMetricAlg($\mathcal{D}$) for metric k-EC-ND

1: **let** network $S \leftarrow \emptyset$.
2: **for** each terminal pair s_i that arrives **do**
3: **let** $\mathcal{E}_i$ denote the set of edges bought by online Alg when we give it demand vertex s_i.
4: **for** each edge $e = (u, v) \in \mathcal{E}_i$ **do**
5: **let** $S \leftarrow S \cup \{(u, x) \mid x \in N_u\} \cup \{(v, x) \mid x \in N_v\}$
6: **let** $S \leftarrow S \cup$ *min-cost matching between* $(N_u \setminus N_v)$ *and* $(N_v \setminus N_u)$
7: **end for**
8: **end for**

Theorem 4.1 *For any set of demands $\mathcal{D}$ that arrive, the network $S_{\mathcal{D}}$ output by algorithm* OnlineMetricAlg *is feasible to k-edge-connecting the root with the demands in $\mathcal{D}$, and has cost $c(S_{\mathcal{D}})$ at most $O(\log n)\mathsf{OPT}_{\mathcal{D}}$, where $\mathsf{OPT}_{\mathcal{D}}$ is the cost of an optimal offline subgraph which k-edge-connects vertices in $\mathcal{D} \cup \{r\}$.*

Proof. We first show that the network $S_{\mathcal{D}}$ is indeed a feasible solution. Let the terminals in $\mathcal{D}$ be $s_1, s_2, \ldots, s_i$, indexed by their arrival times. For any $j \in [1, i]$, let $F_j = \cup_{l=1}^{j} \mathcal{E}_l$ denote the Steiner subgraph bought by Alg in Step 3. Now consider some demand vertex $s_j \in \mathcal{D}$. Since Alg is an online algorithm for Steiner tree, the subgraph F_j contains a path from s_j to r. Consider vertices u and v such that $(u, v) \in F_j$. Since we connect u to N_u and v to N_v and add in a perfect matching between the vertices of $(N_u \setminus N_v)$ and $(N_v \setminus N_u)$ in S, it is easy to see that u and v are k-edge-connected. Therefore, from the transitivity of edge-connectivity, we see that any two vertices that are connected in F_j are in fact k-edge-connected, and hence $S_{\mathcal{D}}$ is a feasible solution.

To bound the cost, we use the following lower bounds on the cost of an optimal solution (similar bounds were also used by Cheriyan and Vetta [CV05] for node-connectivity SNDP):

- $c(\mathsf{OPT}_{\mathcal{D}}) \geq \frac{1}{2} \sum_{v \in D} c(v, N_v)$, and
- $c(F_i) \leq \frac{O(\log n)}{k} c(\mathsf{OPT}_{\mathcal{D}})$.

Let us explain why these are true: In any feasible solution, each terminal has to connect to at least k distinct neighbors. So if we add up the cost of some k outgoing edges from each vertex, the sum should be at most $2c(\mathsf{OPT}_{\mathcal{D}})$ since we can include each edge in $\mathsf{OPT}_{\mathcal{D}}$ at most twice. This gives us the first bound, since $c(v, N_v)$ is at most the sum of the costs on *some* k outgoing edges. For the second bound, consider the optimal fractional solution to the LP relaxation for the rooted k-EC-ND problem on demand set $\mathcal{D}$; clearly the cost of the fractional solution is at most $c(\mathsf{OPT}_{\mathcal{D}})$. Now, if we scale the fractional solution by a factor of k, we obtain a feasible fractional solution for the Steiner tree LP on demand set $\mathcal{D} \cup \{r\}$ of cost at most $\frac{1}{k} c(\mathsf{OPT}_{\mathcal{D}})$. But now because the LP for the minimum cost Steiner tree problem (which is a special case of SNDP) has a constant integrality gap, we get that the cost of an optimal offline Steiner tree feasible to the demand set $\mathcal{D}$ is at most $\frac{2}{k} c(\mathsf{OPT}_{\mathcal{D}})$. The second inequality then follows as a consequence of the $O(\log n)$-competitiveness of the online Steiner tree algorithm Alg.

The total cost of the subgraph $S_{\mathcal{D}}$ is then

$$\begin{aligned} c(S_{\mathcal{D}}) &\leq \textstyle\sum_{u \in F_i} c(u, N_u) + \sum_{(u,v) \in F_i} (c(u, N_u) + c(v, N_v) + k \cdot c(u, v)) \\ &\leq O(1) \textstyle\sum_{u \in F_i} c(u, N_u) + k \sum_{(u,v) \in F_i} c(u, v) \\ &\leq O(\log n)\, c(\mathsf{OPT}_{\mathcal{D}}) \end{aligned}$$

Here, the cost of the min-cost matching between N_u and N_v was bounded by $c(u, N_u) + c(v, N_v) + k \cdot c(u, v)$ by using the triangle inequality of the metric space. Also, the second inequality follows from the assumption that Alg always maintains a bounded-degree online Steiner tree. ∎

It now remains to show how to design the bounded-degree $O(\log n)$-competitive online algorithm for Steiner tree; we now consider this sub-problem.

4.1.1 Online Degree-Bounded Steiner Tree

Imase and Waxman [IW91] show that the greedy algorithm (of each new demand connecting to its nearest vertex on the current solution) is $O(\log n)$-competitive for the online Steiner tree problem. The only problem is that if several terminals share a common vertex as their nearest neighbors, the common vertex would have a very high degree in the Steiner tree we maintain. To avoid this, the simple idea we use is to maintain a *chain* for each vertex v, and connect these terminals (which would have otherwise connected to the common vertex v) to the end of the chain instead. Because the graph is a complete metric, this would allow us to bound the cost of the chain by the cost of the star around each vertex v, and hence let us maintain a low-cost degree bounded solution. In the following, let GreedySteiner denote the online greedy algorithm for the Steiner tree problem.

Algorithm 3 LowDegAlg($\mathcal{D}$) for bounded-degree online Steiner tree

1: **let** network $S \leftarrow \emptyset$; **define** a chain $C_v \leftarrow \emptyset$ for each $v \in V$.
2: **for** each terminal vertex s_i that arrives **do**
3: **let** (v, s_i) denote the edge bought by GreedySteiner when we give it demand vertex s_i.
4: **if** $C_v = \emptyset$ **then**
5: **set** $S \leftarrow S \cup (v, s_i)$; add s_i to the chain C_v
6: **else**
7: **let** v' be the tail of the chain C_v; **set** $S \leftarrow S \cup (v', s_i)$ and **add** s_i to the end of the chain C_v.
8: **end if**
9: **end for**

Theorem 4.2 *The network $S_{\mathcal{D}}$ output by algorithm* LowDegAlg *is feasible to the online Steiner tree problem on demand set $\mathcal{D}$, and has cost $c(S_{\mathcal{D}})$ at most $O(1)c(\text{GreedySteiner}(\mathcal{D}))$. Furthermore, the degree of each vertex in $S_{\mathcal{D}}$ is at most 3.*

Proof. We first show that the algorithm outputs a feasible solution. Consider a newly arrived vertex s_i, and let (v, s_i) denote the edge bought by the online algorithm GreedySteiner (recall that the greedy online Steiner tree algorithm buys only the edge to the nearest neighbor on the current solution). If the chain C_v is empty when s_i arrives, our online algorithm LowDegAlg also buys the edge (v, s_i) and therefore connects s_i to the root r (since v was already connected to the root). If on the other hand C_v was non-empty, then let $C_v \equiv \{v_1, v_2, \ldots, v_t\}$, with the vertices ordered by time of addition to the chain. Then, by the way LowDegAlg adds edges within a chain (in step 7), we are guaranteed that the edges $(v, v_1), (v_1, v_2), \ldots, (v_{t-1}, v_t)$ are all present in the network $S_{\mathcal{D}}$ maintained. And since the edge (v_t, s_i) is also added to $S_{\mathcal{D}}$ when s_i is added to the end of the chain C_v, we see that s_i is connected to v, and therefore to r in $S_{\mathcal{D}}$. This proves the feasibility.

To bound the cost, consider a vertex v, and let the current chain be $C_v \equiv \{v_1, v_2, \ldots, v_t\}$. Now, by the triangle inequality, the total cost of all the edges $(v, v_1), (v_1, v_2), \ldots, (v_{t-1}, v_t)$ can be bounded by $2\sum_{i=1}^{t} c(v, v_i)$, which is precisely twice the total cost incurred by GreedySteiner when connecting $v_1, v_2, \ldots, v_t$ to the vertex v. Therefore, the total cost of $S_{\mathcal{D}}$ can be bounded by $2c(\text{GreedySteiner}(\mathcal{D}))$.

It is also easy to see that the degree of each vertex in $S_{\mathcal{D}}$ is at most 3. To see why, let us consider a vertex v and look at when edges incident at v are added in $S_{\mathcal{D}}$. There are two reasons why such edges are added: **(i)** an edge (v, v') is added when v' arrives to the chain that v is present in, and **(ii)** v can have a chain of its own. In the former case, we know that v belongs to the chain of only one other vertex v' (the vertex to which it was connected in GreedySteiner when it arrived), and therefore v can have at most 2 incident edges by being

present in $C_{v'}$. In the latter case, since it is the head, it can have at most one edge incident on it (with the vertex which first entered v's chain).

Therefore, the network $S_{\mathcal{D}}$ constructed is feasible to the online Steiner tree, has a cost competitive with GreedySteiner, and the degree of each vertex is bounded by 3. ■

4.2 Stochastic and Rent-or-Buy k-EC-ND on Metric Graphs

We now consider the stochastic and rent-or-buy versions of k-EC-ND on complete metrics. We give an $O(1)$-approximation algorithm for metric-k-EC-ND and then show that it admits $O(1)$-strict *cost-shares* (see Section 1.2 for the definition), implying constant-approximations for the metric stochastic and rent-or-buy versions. While better algorithms are known for k-EC-ND (and even the more general k-vertex-connectivity problem [CV05]), we present an approximation algorithm which can be shown to admit good cost-shares.

4.2.1 Overview

We first give a brief overview of how the framework of *boosted sampling* [GPRS04] can be applied to obtain approximation algorithms for stochastic (and rent-or-buy) versions of network design problems. Given an instance of a two-stage stochastic network design problem (where different terminal pairs have *independent* arrival probability distributions), the boosted sampling framework proceeds as follows:

(i) Sample each terminal pair according to its probability of being present (boosted by a scaling factor of σ that depends on the inflation parameter).
(ii) In the first stage, build an approximate Steiner forest S connecting the sampled terminal pairs (using a suitable approximation algorithm Alg).
(iii) In the second phase, if a terminal pair $\{s_i, t_i\}$ is not connected by S, connect the end vertices s_i and t_i by a shortest augmenting path w.r.t S.

To prove that this algorithm indeed produces a constant approximation w.r.t the optimal strategy, the main technical tool used is that of *strict cost-sharing schemes* (refer to Section 1.2 for a formal definition). Informally, these cost-shares (which are coupled with the algorithm Alg) are designed to make sure that the augmenting cost is small over all possible terminal pairs (which is useful in bounding the second stage cost).

For the Steiner forest problem, Gupta et al. [GPRS04] designed a 4-approximation algorithm that admit $O(1)$-strict cost-sharing schemes, thereby giving $O(1)$-approximation algorithms for the stochastic (and rent-or-buy) versions. Subsequently, Fleischer et al. [FKLS06] showed that the primal-dual AKR algorithm for Steiner forest admits $O(1)$-strict cost-shares via an elegant proof associating witness terminals for edges bought by the algorithm.

Following our approach for the online k-EC-ND algorithm for metric instances in Section 4.1, we extend the AKR algorithm to get an approximation algorithm for k-EC-ND that admits $O(1)$-strict cost-shares. Just like the issue that arose for the online algorithm, the main hurdle in directly extending the AKR algorithm (and the cost-sharing scheme) is in modifying the AKR solution to ensure that the degree of any vertex is small. While this as such is trivial (we can perform an Euler tour of the AKR solution), the crux of the argument is to show that this modified algorithm also admits $O(1)$-strict cost-shares. In the following section, we first formally define the k-EC-ND approximation algorithm and then explain our cost-sharing mechanism.

4.2.2 Constant-Factor Approximation for Metric k-EC-ND

Consider an instance $G = (V, E)$ of k-EC-ND with terminal pairs in $\mathcal{D}$; let D represent the set of all terminals, i.e., $D = \cup_{\{s_i,t_i\}\in\mathcal{D}}\{s_i, t_i\}$. Call a set $S \subseteq V$ *valid* if there exist a demand $\{s_i, t_i\} \in \mathcal{D}$ such that $|S \cap \{s_i, t_i\}| = 1$. Define ∂S to be the set of edges with one endpoint in S, and $x(E') = \sum_{e\in E'} x_e$. Finally, let N_v represent

the set of the k nearest neighbors of vertex v in G. The LP relaxation of the k-EC-ND problem is the following:

$$\begin{array}{lll} (LP_k) & \text{minimize} & \sum_{e\in E} c_e x_e \\ & \text{subject to} & (1) \quad x(\partial S) \geq k \qquad \forall \text{valid } S \subseteq V \\ & & (2) \quad 0 \leq x_e \leq 1, \qquad \forall e \in E \end{array}$$

Let OPT and $\mathsf{OPT}_{\mathsf{LP}}$ be optimal integral and fractional solutions to the given instance; clearly $c(\mathsf{OPT}_{\mathsf{LP}}) \leq c(\mathsf{OPT})$. Our algorithm follows the ideas used in the online algorithm, with the following changes: instead of running the online algorithm for Steiner tree, we run the AKR algorithm ([AKR95]) for Steiner forest to 1-edge-connect the demand pairs in the first step. The AKR algorithm is a primal-dual 2-approximation algorithm for the Steiner forest problem—when given a set of terminal pairs $\mathcal{D}$, it outputs a Steiner forest of cost at most *twice* the cost of an optimal fractional solution to the standard LP relaxation of Steiner forest (which is just the above LP relaxation LP_k with $k = 1$).

Getting back to our algorithm, in our second step, in order to get a low-degree Steiner forest, we simply take an Euler tour of the AKR solution. Finally, we k-edge-connect u and v (using nearby neighbors N_u and N_v) for any edge (u, v) that the AKR algorithm buys, just like in the online algorithm.

Algorithm 4 MetricAlg($\mathcal{D}$) for metric k-EC-ND

1: **let** network $S \leftarrow \emptyset$. Run the AKR algorithm on $\mathcal{D}$ to get forest F.
2: **let** $\widetilde{F} \leftarrow$ subgraph obtained by taking Euler tour of each component of F.
3: **for** each edge $e = (u, v)$ in $\widetilde{F}$ **do**
4: **let** $S \leftarrow S \cup \{(u, x) \mid x \in N_u\} \cup \{(v, x) \mid x \in N_v\}$
5: **let** $S \leftarrow S \cup$ *min-cost matching between* N_u *and* N_v
6: **end for**

Theorem 4.3 *The network S output by algorithm* MetricAlg *k-edge-connects the terminal pairs in $\mathcal{D}$, and has cost $c(S) \leq 10\, c(\mathsf{OPT})$. Furthermore, if $(u, v) \in F$, then u and v are k-edge-connected in S.*

Proof. The proof is very similar to that of Theorem 4.1. We begin by showing that network S is indeed a feasible solution. Consider a terminal pair $\{s, t\} \in \mathcal{D}$: since F is a feasible Steiner forest solution for $\mathcal{D}$, we know that s and t belong to the same tree in F and therefore, to the same cycle in $\widetilde{F}$. We now show that any pair of vertices that lie on a cycle in $\widetilde{F}$ are k-edge-connected in S. Consider vertices u and v such that $(u, v) \in \widetilde{F}$. Since we connect u to N_u and v to N_v and add in a perfect matching between the vertices of $N_u \setminus N_v$ and $N_v \setminus N_u$ in S, it is easy to see that u and v are k-edge-connected. By transitivity of edge-connectivity, any two vertices on a cycle in $\widetilde{F}$ are k-edge-connected. This proves that S is a feasible solution. To bound the cost, we again use the following lower bounds (almost identical to the ones used in the online algorithm):

- $c(\mathsf{OPT}) \geq \frac{1}{2} \sum_{v\in D} c(v, N_v)$, and
- $c(\widetilde{F}) \leq \frac{4}{k}\, c(\mathsf{OPT})$.

For completeness, let us explain why these are true: The first bound has already been established in the proof for the online algorithm in the earlier section (proof of Theorem 4.1). For the second bound, consider the optimal fractional solution to the LP relaxation for the rooted k-EC-ND problem on demand set $\mathcal{D}$; clearly the cost of the fractional solution is at most $c(\mathsf{OPT}_{\mathcal{D}})$. Now, if we scale the fractional solution by a factor of k, we obtain a feasible fractional solution for the Steiner forest LP on demand set $\mathcal{D} \cup \{r\}$ of cost at most $\frac{1}{k} c(\mathsf{OPT}_{\mathcal{D}})$. But now because the AKR for the minimum cost Steiner forest problem has a constant approximation factor (w.r.t the optimal LP solution), we get that the cost of the solution $\mathcal{F}$ is at most $\frac{2}{k} c(\mathsf{OPT}_{\mathcal{D}})$. Making an Euler tour of F to get $\widetilde{F}$ only increases the cost by a factor of at most 2.

The total cost of S is then

$$\begin{aligned} c(S) &\le \textstyle\sum_{u\in\widetilde{F}} c(u,N_u) + \sum_{(u,v)\in\widetilde{F}} (c(u,N_u)+c(v,N_v)+k\cdot c(u,v)) \\ &\le 3\textstyle\sum_{u\in\widetilde{F}} c(u,N_u) + k\sum_{(u,v)\in\widetilde{F}} c(u,v) \\ &\le 10\, c(\mathsf{OPT}) \end{aligned}$$

In the second step, we used the fact that because each vertex has degree 2 in $\widetilde{F}$, the term $c(u,N_u)$ can appear at most twice in the latter summation $\sum_{(u,v)\in\widetilde{F}} (c(u,N_u)+c(v,N_v)+k\cdot c(u,v))$. ■

4.2.3 Getting Strict Cost-Shares

We now show how we can get strict cost-shares for the above algorithm. As the basis for our cost-sharing scheme, we use the cost-shares associated with the AKR algorithm, as given by Fleischer et al. [FKLS06]. We refer to their cost-sharing scheme as the FKLS cost-shares, or the FKLS analysis.

Let $F^{\mathcal{D}}$ denote the AKR solution on demand set $\mathcal{D}$. The FKLS analysis defines the cost-sharing functions $\xi' : E \times V \to \mathbb{R}$ and $\xi : \mathcal{D} \to \mathbb{R}$ in the following manner:

(i) Each edge $e \in F^{\mathcal{D}}$ is assigned two *witness* terminals w_1 and w_2 such that $\xi'(e,w_1) = \xi'(e,w_2) = c_e/4$. The function $\xi'(e,v)$ is set to 0 for all $v \in V \setminus \{w_1,w_2\}$.
(ii) For any vertex u and an edge e not in $F^{\mathcal{D}}$, $\xi'(e,u)$ is set to 0.
(iii) The total cost-share of a terminal pair $\{s_i,t_i\}$ is then defined as $\xi(\{s_i,t_i\}) = \sum_{e\in F^{\mathcal{D}}} (\xi'(e,s_i)+\xi'(e,t_i))$.

In further notation, for any edge $e \in F^{\mathcal{D}}$, let τ_e denote the time at which e was bought by the AKR algorithm; also denote the time at which $\{s_i,t_i\}$ gets connected in $F^{\mathcal{D}}$ by τ_i. The FKLS analysis shows that the witnesses satisfy the following properties:

1. Consider a demand $\{s_i,t_i\}$ and any edge $e \in F^{\mathcal{D}}$ bought at time $\tau_e \le \tau_i$. If neither s_i nor t_i is a witness for e, then e is also bought in the run of AKR $(\mathcal{D} \setminus \{s_i,t_i\})$. In particular, for any edge e on the unique path connecting s_i and t_i in $F^{\mathcal{D}}$, if s_i and t_i don't witness e, then $e \in F^{\mathcal{D}\setminus\{s_i,t_i\}}$.

2. $\sum_{i=1}^{|\mathcal{D}|} \xi(\{s_i,t_i\}) \le \frac{2}{k} c(\mathsf{OPT}_{\mathsf{LP}}(\mathcal{D}))$. Further, the solution obtained by running AKR on $\mathcal{D} \setminus \{s_i,t_i\}$ can be augmented with edges of cost $O(1)\xi(\{s_i,t_i\})$ to get a subgraph which connects s_i and t_i. In fact, these "augmenting" edges are those which s_i or t_i witness.

Given that the 1-edge-connectivity problem has nice witness properties, the most natural thing to try would be to define cost-shares for k-edge-connectivity in the following way: for any edge $e = (u,v)$ that s_i or t_i witness, the cost-share for $\{s_i,t_i\}$ includes the cost of k-edge-connecting u and v (at most $2\,c(u,N_u) + 2\,c(v,N_v) + k\,c(u,v)$). When defined in this form, although we would be able to augment a solution of MetricAlg$(\mathcal{D} \setminus \{s_i,t_i\})$ to k-edge-connect s_i-t_i by paying $O(1) \times \xi^k(\{s_i,t_i\})$, we cannot directly bound $\sum_{i=1}^{|\mathcal{D}|} \xi^k(\{s_i,t_i\})$, since the quantity $c(u,N_u)$ could be counted several times. However, this can happen only if the degree of u in the approximate solution is high (just like in the online algorithm). We therefore look at transforming the AKR solution into a low-degree one while *preserving* the witness properties. (An Euler tour would get us the low-degree tree, but it would not satisfy *good* witness properties we desire.)

Let $F^{\mathcal{D}}$ be the forest obtained by running the AKR algorithm on demand set $\mathcal{D}$. We apply the following modification step for each tree in $F^{\mathcal{D}}$. Consider a tree $T^{\mathcal{D}}$, and arbitrarily root it at r. We now perform a reverse breadth-first (bottom up) traversal, and create a modified solution F^{mod} (which is set to $\emptyset$ initially).

Modification: Suppose we are at vertex u in our traversal: Nothing is done if it is a leaf. If it is an internal node having degree 2, then the edge between u and it's child is included in F^{mod}. If it has degree more than 2,

then we perform the following local modification (u is said to be the *main vertex* being *altered* in the step, and the edges being altered are the children edges incident at u):

Let $v_1, v_2, \ldots, v_p$ be an ordering of the child vertices of u ordered such that $\tau_{(u,v_1)} \leq \tau_{(u,v_2)} \leq \ldots \leq \tau_{(u,v_p)}$. We anchor the edge (u, v_1) and add it to F^{mod}. For every other edge (u, v_i), we add the edge (v_i, v_{i-1}) to F^{mod}. The witnesses of (u, v_1) remain the same as those assigned by the FKLS algorithm, and the witnesses of the edge (v_i, v_{i-1}) in F^{mod} are the FKLS witnesses of (u, v_i) in $F^{\mathcal{D}}$. Note that the degree of u in F^{mod} is reduced to 2, whereas the degree of u's child vertices (which were 2 before this step) are increased by at most 1. After this step, u would never be the main vertex being altered in any step meaning that it's degree will be at most 3 in F^{mod}.

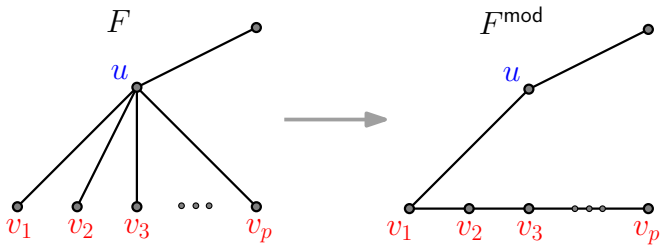


Figure 4.3: A step in the modification: u is the main vertex being altered

This local modification is performed in a reverse breadth first fashion. It is easy to see that we obtain a forest whose cost is at most twice the cost of the AKR solution. Further, each vertex has degree at most 3 and each edge has at most 2 witnesses.

Lemma 4.4 *The forest F^{mod} created by the above modification is such that the cost $c(F^{\mathsf{mod}})$ is at most $2c(F^{\mathcal{D}})$, and any vertex has degree at most 3 in F^{mod}. Furthermore, there exists witness definitions such that the following properties hold.*

(i) If W_i is the set of edges in F^{mod} for which either s_i or t_i is a witness, then for any edge (u, v) in the unique path connecting s_i and t_i in $F^{\mathcal{D}}$, u and v remain connected in the subgraph $W_i \cup F^{\mathcal{D}\setminus\{s_i,t_i\}}$, where $F^{\mathcal{D}\setminus\{s_i,t_i\}}$ is the forest returned by $\mathsf{AKR}(\mathcal{D} \setminus \{s_i, t_i\})$.
(ii) At most 2 terminals witness any edge in F^{mod}.

Proof. The cost and degree bound follow directly as a consequence of the way our modification algorithm worked. We now prove the witness properties by showing that u and v are in fact connected by a path comprising of a sequence of edges in W_i followed by an edge which belongs to $F^{\mathcal{D}\setminus\{s_i,t_i\}}$. Consider the stage in the alteration procedure when (u, v) is being altered. One of u or v has to be the main node being altered. Without loss of generality, we assume that u is the vertex being altered. Two cases are to be considered:
Case (1): (u, v) is not witnessed by $\{s_i, t_i\}$: Since (u, v) lies on the unique s_i-t_i path in $F^{\mathcal{D}}$, we know that $\tau_{(u,v)} \leq \tau_i$. Therefore, by the first property of the FKLS analysis, this edge will be bought by the AKR algorithm when run on $\mathcal{D} \setminus \{s_i, t_i\}$, and therefore u and v are connected in $F^{\mathcal{D}\setminus\{s_i,t_i\}} \subseteq W_i \cup F^{\mathcal{D}\setminus\{s_i,t_i\}}$.
Case (2): (u, v) is witnessed by one of $\{s_i, t_i\}$. Recall that we had assumed that u is the main vertex being altered. In the case that (u, v) was the edge being anchored, we know that (u, v) is present in the modified tree F^{mod} and has the same witnesses as before, meaning $(u, v) \in W_i \subseteq W_i \cup F^{\mathcal{D}\setminus\{s_i,t_i\}}$. If (u, v) was not the edge being anchored, let $v_1, v_2, \ldots, v_p$ be the ordering of the child vertices of u chosen by the alteration procedure. Note that $v \in \{v_2, v_3, \ldots, v_p\}$. Without loss of generality, let v be v_q. Also, let r be the largest index such that $1 \leq r < q$ and that (u, v_r) is not witnessed by either s_i or t_i. There are two cases to be considered.

- If such an r does not exist: it means that each of the edges $(u, v_1), (u, v_2), \ldots, (u, v_q)$ are witnessed

by s_i or t_i, and therefore $(u, v_1), (v_1, v_2), \ldots, (v_{q-1}, v_q)$ all belong to W_i, meaning that u and v are connected in $W_i \cup F^{\mathcal{D}\setminus\{s_i,t_i\}}$.

- If such an r exists: we know that each of the edges $(u, v_{r+1}), \ldots, (u, v_q)$ are witnessed by s_i or t_i—this means that each of the edges $(v_r, v_{r+1}), \ldots, (v_{q-1}, v_q)$ are in W_i. Further, by the way we ordered the children of u, it is clear that $\tau_{(u,v_r)} \leq \tau_{(u,v_q)} \leq \tau_i$. The latter inequality is because of the fact that (u, v_q) is on the unique path connecting s_i and t_i, and therefore cannot be bought after s_i and t_i are connected. Hence, by property 1 of the FKLS algorithm, we know that (u, v_r) is bought in the run of $\mathsf{AKR}(\mathcal{D} \setminus \{s_i, t_i\})$. Therefore, u and v are connected in $W_i \cup F^{\mathcal{D}\setminus\{s_i,t_i\}}$.

This proves the desired witness properties, and hence completes the proof. ■

We are now ready to define the $O(1)$-strict cost-shares for this problem.

Cost-Shares: For each terminal pair $\{s_i, t_i\}$, we set its cost-share to be

$$\xi^k(\{s_i, t_i\}) = \sum_{(u,v)\in W_i} (2\, c(u, N_u) + 2\, c(v, N_v) + k\, c(u, v))$$

Recall that W_i is the set of edges which either s_i or t_i witness in F^{mod}. Since each vertex in F^{mod} has degree at most 3 and each edge e has at most 2 witnesses, we get $\sum_i \xi^k(\{s_i, t_i\}) \leq \sum_{u\in D} 12\, c(u, N_u) + 2k \sum_{e\in F^{\mathsf{mod}}} c(u, v) \leq 32\, c(\mathsf{OPT}(\mathcal{D}))$. To show that these cost-shares are $O(1)$-strict, we also need to give an algorithm which can augment edges of cost $\xi^k(\{s_i, t_i\})$ to a subgraph returned by $\mathsf{MetricAlg}(\mathcal{D} \setminus \{s_i, t_i\})$ in order to k-edge-connect s_i and t_i.

Augmentation Algorithm: $\mathsf{Augment}\ (\{s_i, t_i\})$: For all $(u, v) \in W_i$, buy the set of edges of minimum cost to k-edge-connect u and v.

Analysis: From Lemma 4.4, we know that if an edge (u, v) lies on the unique path connecting s_i and t_i in $F^{\mathcal{D}}$, then u and v are connected in $W_i \cup F^{\mathcal{D}\setminus\{s_i,t_i\}}$. Now consider any edge $(u', v') \in W_i \cup F^{\mathcal{D}\setminus\{s_i,t_i\}}$. If (u', v') is in W_i, then the augmentation algorithm would k-edge-connect the vertices u' and v'. If it is in $F^{\mathcal{D}\setminus\{s_i,t_i\}}$, then from Theorem 4.3, $\mathsf{MetricAlg}(\mathcal{D} \setminus \{s_i, t_i\})$ would k-edge-connect u' and v'. Hence, each edge (u', v') on the $u - v$ path contained in $W_i \cup F^{\mathcal{D}\setminus\{s_i,t_i\}}$ is such that u' and v' are k-edge-connected in $\mathsf{Augment}(\{s_i, t_i\}) \cup \mathsf{MetricAlg}(\mathcal{D} \setminus \{s_i, t_i\})$. Therefore, from transitivity of edge-connectivity, we get that s_i and t_i are k-edge-connected in $\mathsf{Augment}(\{s_i, t_i\}) \cup \mathsf{MetricAlg}(\mathcal{D} \setminus \{s_i, t_i\})$. This, and the fact that the cost of the augmenting edges to k-edge-connect s_i-t_i is at most $\xi^k(\{s_i, t_i\})$, establish the $O(1)$-strict cost-shares for Algorithm MetricAlg. The following theorem therefore follows.

Theorem 4.5 *The algorithm* MetricAlg *permits $O(1)$-strict cost-shares for* k-EC-ND*, implying $O(1)$ approximations for metric rent-or-buy and two-stage stochastic (with independent arrivals)* k-EC-ND.

Acknowledgments. We thank Chandra Chekuri, Amit Kumar and Stefano Leonardi for useful discussions. We also thank anonymous reviewers of earlier versions of this paper for helpful comments.

References

[AAA+03] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Seffi Naor. The online set cover problem. In *35th STOC*, pages 100–105, 2003.

[AAA+04] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Seffi Naor. A general approach to online network optimization problems. In *15th SODA*, pages 577–586, 2004.

[AAB04] Baruch Awerbuch, Yossi Azar, and Yair Bartal. On-line generalized Steiner problem. *Theoret. Comput. Sci.*, 324(2-3):313–324, 2004.

[ABN08] Ittai Abraham, Yair Bartal, and Ofer Neiman. Nearly tight low stretch spanning trees. In *48th FOCS*, pages 781–790, 2008.

[AKR95] Ajit Agrawal, Philip Klein, and R. Ravi. When trees collide: an approximation algorithm for the generalized Steiner problem on networks. *SIAM J. Comput.*, 24(3):440–456, 1995. (Preliminary version in *23rd STOC*, 1991).

[Bar96] Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *37th FOCS*, pages 184–193, 1996.

[BC97] Piotr Berman and Chris Coulston. On-line algorithms for steiner tree problems (extended abstract). In *29th STOC*, pages 344–353, 1997.

[BKN08] Nikhil Bansal, Rohit Khandekar, and Viswanath Nagarajan. Additive guarantees for degree bounded directed network design. In *40th STOC*, pages 769–778, 2008.

[BN06] Niv Buchbinder and Joseph (Seffi) Naor. Improved bounds for online routing and packing via a primal-dual approach. In *46th FOCS*, pages 293–304, 2006.

[CCK08] Tanmoy Chakraborty, Julia Chuzhoy, and Sanjeev Khanna. Network design for vertex connectivity. In *40th STOC*, pages 167–176, 2008.

[CFLY08] Yuk Hei Chan, Wai Shing Fung, Lap Chi Lau, and Chun Kong Yung. Degree bounded network design with metric costs. In *48th FOCS*, 2008.

[CGK$^+$08] Chandra Chekuri, Anupam Gupta, Nitish Korula, Ravishankar Krishnaswamy, and Amit Kumar. Cost-sharing for subgraph k-edge-connectivity. unpublished, July 2008.

[CK08a] Chandra Chekuri and Nitish Korula. Single-sink network design with vertex connectivity requirements. In *FST&TCS*, 2008.

[CK08b] Julia Chuzhoy and Sanjeev Khanna. Algorithms for single-source vertex connectivity. In *48th FOCS*, pages 105–114, 2008.

[CK09] Julia Chuzhoy and Sanjeev Khanna. An o($k^3 \log n$)-approximation algorithm for vertex-connectivity survivable network design. In *49th FOCS*, pages 437–441, 2009.

[CV05] Joseph Cheriyan and Adrian Vetta. Approximation algorithms for network design with metric costs. In *37th STOC*, pages 167–175, 2005.

[CVV03] Joseph Cheriyan, Santosh Vempala, and Adrian Vetta. An approximation algorithm for the minimum-cost k-vertex connected subgraph. *SIAM J. Comput.*, 32(4):1050–1055, 2003.

[EEST05] Michael Elkin, Yuval Emek, Daniel A. Spielman, and Shang-Hua Teng. Lower-stretch spanning trees. In *37th STOC*, pages 494–503, 2005.

[FJW06] Lisa Fleischer, Kamal Jain, and David P. Williamson. Iterative rounding 2-approximation algorithms for minimum-cost vertex connectivity problems. *J. Comput. System Sci.*, 72(5):838–867, 2006.

[FKLS06] Lisa Fleischer, Jochen Könemann, Stefano Leonardi, and Guido Schäfer. Simple cost sharing schemes for multicommodity rent-or-buy and stochastic steiner tree. In *38th STOC*, pages 663–670, 2006.

[FL08] Jittat Fakcharoenphol and Bundit Laekhanukit. An O($\log^2$ k)-approximation algorithm for the k-vertex connected spanning subgraph problem. In *40th STOC*, pages 153–158, 2008.

[FRT04] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. System Sci.*, 69(3):485–497, 2004.

[GGP+94] Michel X. Goemans, Andrew V. Goldberg, Serge Plotkin, David B. Shmoys, Éva Tardos, and David P. Williamson. Improved approximation algorithms for network design problems. In *5th SODA*, pages 223–232, 1994.

[GK11] Anupam Gupta and Jochen Konemann. Approximation algorithms for network design: A survey. *Surveys in Operations Research and Management Science*, 16(1):3 – 20, 2011.

[GKPR03] Anupam Gupta, Amit Kumar, Martin Pál, and Tim Roughgarden. Approximations via cost-sharing. In *44th FOCS*, pages 606–615, 2003.

[GKPR07] Anupam Gupta, Amit Kumar, Martin Pál, and Tim Roughgarden. Approximation via cost sharing: simpler and better approximation algorithms for network design. *J. ACM*, 54(3):Art. 11, 38 pp. (electronic), 2007.

[GPRS04] Anupam Gupta, Martin Pál, R. Ravi, and Amitabh Sinha. Boosted sampling: Approximation algorithms for stochastic optimization problems. In *36th STOC*, pages 417–426, 2004.

[GW95] Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995.

[IKMM04] Nicole Immorlica, David Karger, Maria Minkoff, and Vahab Mirrokni. On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems. In *15th SODA*, pages 684–693, 2004.

[IW91] Makoto Imase and Bernard M. Waxman. Dynamic Steiner tree problem. *SIAM J. Discrete Math.*, 4(3):369–384, 1991.

[Jai01] Kamal Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001. (Preliminary version in *39th FOCS*, pages 448–457, 1998).

[KKL04] Guy Kortsarz, Robert Krauthgamer, and James R. Lee. Hardness of approximation for vertex-connectivity network design problems. *SIAM J. Comput.*, 33(3):704–720, 2004.

[KN03] Guy Kortsarz and Zeev Nutov. Approximating node connectivity problems via set covers. *Algorithmica*, 37(2):75–92, 2003.

[KR93] Philip N. Klein and R. Ravi. When cycles collapse: A general approximation technique for constrained two-connectivity problems. In *3rd IPCO*, pages 39–56, 1993.

[KR96] Samir Khuller and Balaji Raghavachari. Improved approximation algorithms for uniform connectivity problems. *J. Algorithms*, 21(2):434–450, 1996.

[LNSS07] Lap Chi Lau, Joseph Naor, Mohammad R. Salavatipour, and Mohit Singh. Survivable network design with degree or order constraints. In *39th STOC*, pages 651–660, 2007.

[LS08] Lap Chi Lau and Mohit Singh. Additive approximation for bounded degree survivable network design. In *40th STOC*, pages 759–768, 2008.

[RS04] R. Ravi and Amitabh Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. In *10th IPCO*, pages 101–115, 2004.

[SS06] David B. Shmoys and Chaitanya Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *J. ACM*, 53(6):978–1012, 2006.

[WGMV95] David P. Williamson, Michel X. Goemans, Milena Mihail, and Vijay V. Vazirani. A primal-dual approximation algorithm for generalized Steiner network problems. *Combinatorica*, 15(3):435–454, 1995.

A Lower Bound

In this section, we show that, if $\mathcal{D}$ is the set of demands that have arrived till some point, then there are instances where the competitive ratio of any online algorithm is $\Omega(|\mathcal{D}|)$ for $|\mathcal{D}| = O(\log n)$, even for the rooted 2-edge connectivity problem. Consider the graph given in Figure A.4. There is a binary tree of depth L, and all the leaves are connected to the root with distinct *"back" edges*. All edges in this graph have unit cost. For ease of exposition, we will assume that the edges bought when seeing any demand are a minimal set of edges to achieve the connectivity requirement for that demand; any edges not in the minimal set are considered to be bought at the first time they are actually used.

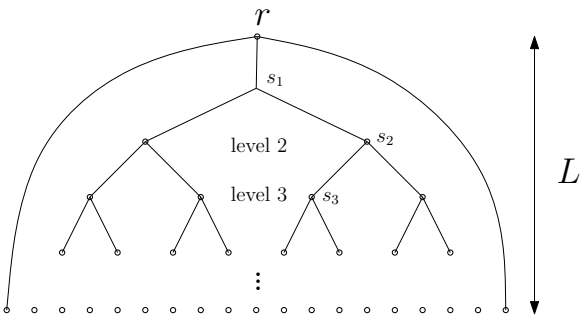


Figure A.4: A lower bound of $\Omega(L)$

All the requests will be vertices that need 2-connectivity to the root r. The first request is level-1 vertex s_1; one feasible solution is to buy the edge s_1-r, and the second path is some path from s_1 to a leaf and back to r using a "back" edge. However, this is not the only minimal solution possible: perhaps the algorithm can buy two disjoint paths from s_1 to two leaves which use their back edges to connect to r. In any case, there will be at least one vertex on level 3 that is not yet connected to the root. We then give any such vertex on level 3 as the next request. The third request will be some vertex on level 5 that is a descendant of the second request, and which is not yet connected to the root; in general, the next request is always chosen to be a descendant of the previous demands. This ensures that there is always a feasible solution of cost $L + 1$ for all the demands seen thus far, whereas the online algorithm pays at least $\Omega(L)$ for the first $\Omega(L)$ requests, giving us the claimed lower bound.

Note that this construction also works against oblivious adversaries if we choose a random descendant at level $(2i - 1)$ as the i^{th} request.