Online and Stochastic Survivable Network Design

Anupam Gupta[*]
Computer Science Dept.
Carnegie Mellon University
Pittsburgh PA 15213
anupamg@cs.cmu.edu

Ravishankar Krishnaswamy[*]
Computer Science Dept.
Carnegie Mellon University
Pittsburgh PA 15213
ravishan@cs.cmu.edu

R. Ravi[†]
Tepper School of Business
Carnegie Mellon University
Pittsburgh PA 15213
ravi@cmu.edu

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 $O(r_{\max} \log^3 n)$ competitive online algorithm for this edge-connectivity network design problem, 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.

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. For these two problems, 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 our results to give $O(1)$-strict cost shares, which gives constant-factor rent-or-buy and stochastic algorithms for these instances.

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems

General Terms: Algorithms, Theory

Keywords: Approximation Algorithms, Online Algorithms, Network Design Problems

[*]Supported in part by NSF awards CCF-0448095 and CCF-0729022, and an Alfred P. Sloan Fellowship.
[†]Supported in part by NSF grant CCF-0728841.

1. INTRODUCTION

We consider the edge-connectivity version of the *survivable network design* problem (SNDP): given a graph $G = (V, E)$ with 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), it has been widely studied from the viewpoint of approximation algorithms. 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 [22]. Subsequently, the first use 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) [28].

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]. Now a sequence of vertex pairs $(i, j) \in V \times V$ is presented to us over time, each 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 r_{ij} edge-connectivity 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 rooted 2-connectivity case (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, one can show lower bounds of $\Omega(\min\{|\mathcal{D}|, \log n\})$ on the competitive ratio for 2-connectivity if $\mathcal{D}$ is the set of terminal pairs given to the algorithm (see Appendix A).

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

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

[1]We can remove this assumption of knowing $r_{\max}$ up front by losing another $\log r_{\max}$ in our approximation guarantee; for simplicity we assume that we know $r_{\max}$.

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 is polynomial-time only for constant $r_{\max}$. Please read Section 2 for a high-level overview of our ideas.

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 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'll demand k-connectivity. (For simplicity we assume all pairs require the same connectivity k.) We can buy some edges today (at cost $c(e)$), but if we wait for the actual set $\mathcal{D}$ the edges will cost $\lambda c(e)$ tomorrow, for a prespecified inflation parameter $\lambda \geq 1$; we seek to minimize the expected cost. Given an α-approximation algorithm for the basic k-edge-connectivity problem that admits β-strict cost shares, one can get a randomized algorithm which is an $(\alpha + \beta)$-approximation for the stochastic version.

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.

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 [16], and independently (but explicitly) by Chekuri et al. [12] 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 cost-sharing mechanisms for the special case when the underlying graph is complete and the costs satisfy the triangle inequality to get the following result. At a high level, our cost-shares are based on combining ideas from the strict cost-shares of Fleischer et al. [21] for Steiner forest, and the metric k-connectivity algorithms of Cheriyan and Vetta [15].

Theorem 1.3 *There is an $O(1)$-approximation algorithm for metric k-edge-connectivity with $O(1)$-strict cost-shares. Hence, both the problems of rent-or-buy k-edge-connectivity and two-stage stochastic k-edge-connectivity with independent decisions admit constant-factor approximations.*

1.1 Related Work

Steiner network problems have received considerable attention in approximation algorithms: Agrawal et al. [2] and Goemans and Williamson [23] used primal-dual methods to design approximation algorithm for Steiner forests and other 1-connectivity problems (and some higher connectivity problems where multiple copies of edges could be used). Klein and Ravi [30] gave an algorithm for the 2-connectivity problem, which was extended by Williamson et al. [37] and Goemans et al. [22] to higher connectivity problems, yielding $O(\log k)$ approximations for k-connectivity, all using primal-dual methods. Jain [28] gave 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 $\mathsf{P} \neq \mathsf{NP}$) for network design with degree constraints [33, 34, 6]. Vertex connectivity problems are less well-understood: [14, 32, 18] consider problems of *spanning* k-connectivity, and provide approximation algorithms with varying guarantees depending on k. Fleischer et al. [20] give a 2-approximation for vertex connectivity when all $r_{ij} \in \{0, 1, 2\}$. Recently, the papers [10, 13, 16] consider the problem of *single source* k-vertex connectivity, culminating in a simple greedy $O(k \log n)$ algorithm. In fact, the papers [13, 16] also implicitly give $O(k)$-strict cost shares for the single source node-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. When the edges have metric costs, there have, expectedly, been better approximation algorithms for vertex connectivity. Khuller and Raghavachari [29] gave $O(1)$-approximations for k-node connected spanning subgraphs. Cheriyan and Vetta [15] later gave $O(1)$ approximations for the single source k-connected problem and a $O(\log r_{max})$-approximation for metric node-connected SNDP. Recently, Chan et al. [11] give constant factor approximations for several degree bounded problems on metric graphs. As for inapproximability, Kortsarz et al. [31] give $2^{\log^{1-\varepsilon} n}$ hardness results for the node-connected survivable network design problem.

Imase and Waxman [26] first considered the online Steiner tree problem and gave a tight $\Theta(\log k)$-competitive algorithm. Awerbuch, Azar and Bartal [5], and subsequently Berman and Coulston [8] gave the same guarantee for the online Steiner forest problem. We do not see how to use these ideas for higher connectivity. In this paper, we use results of Alon et al. [3] for the online (weighted) set cover problem; the ideas here have been extended by Alon et al. [4] and Buchbinder and Naor [9] to get online primal-dual based algorithms for fractional generalized network design. (We can solve the fractional k-connectivity problems using these, but we do not know how to round them well.)

The use of strict cost-shares to get algorithms for rent-or-buy network design appears in [24]. Approximation algorithms for two-stage stochastic problems were studied in [27, 35], and some general techniques were given by [25, 36]; in particular, using strict cost-shares appears in [25].

1.2 Preliminaries

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 losses in the guarantees.

1.2.1 Strictness

An α-approximation algorithm Alg is said to be β-strict for the k-EC-ND problem if there exist cost shares $\xi(\mathcal{D}, (s_i, t_i))$ for all $(s_i, t_i) \in \mathcal{D}$ (where $\mathcal{D} \subseteq \binom{V}{2}$ is the set of demand pairs) such that:

- $\sum_{(s_i,t_i)\in\mathcal{D}} \xi(\mathcal{D}, (s_i, t_i)) \leq \mathsf{OPT}_{\mathcal{D}}$, where $\mathsf{OPT}_{\mathcal{D}}$ is the cost of the optimal solution.
- There is an efficient augmenting procedure Augment (which takes as input a terminal pair) such that the subgraph $\mathsf{Augment}((s_i, t_i)) \cup \mathsf{Alg}(\mathcal{D}\backslash(s_i, t_i))$ k-connects s_i and t_i, and the cost of edges output by $\mathsf{Augment}(s_i, t_i)$ is at most $\beta\, \xi(\mathcal{D}, (s_i, t_i))$ for all $(s_i, t_i) \in \mathcal{D}$.

1.2.2 Cost Shares from Online Algorithms

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

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-connected s_i-t_i (where $l < k$), and we want to $l + 1$ connect s_i-t_i. If we think of the s_i-t_i cuts as sets, then we want 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 hit the cuts by *paths* connecting two nodes in H (instead of edges in G), but there could be exponentially many such paths, 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 with respect to this spanning tree T. Interestingly, the (random) distance-preserving property allows us to control the cost of these connectivity augmentations. Of course, this high-level view oversimplifies things a bit: please read on for details. In Sections 3.1 and 3.2 we show how we can hit cuts by a small number of cycles/paths, and then Sections 3.3 and 3.4 use these ideas to develop our algorithms.

3. 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 guarantees for the rent-or-buy and stochastic k-EC-ND problems.

3.1 Embedding into Backboned Graphs

One of the major advantages of network design problems which only sought 1-connectivity was that one could embed the underlying metric space into random trees [7, 19, 17, 1], since the problems were easier on trees. Such a reduction seems impossible even for 2-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. [17, 1]. Formally, this gives a random spanning tree $T = (V, E_T \subseteq E)$ of G with edge lengths $\widehat{c}_T$ where for all $x, y \in V$, it holds that

- $\widehat{c}_T(e) = c(e)$ for all edges $e \in E_T$, and hence $d_T(x, y) \geq d_G(x, y)$
- $\mathbb{E}[d_T(x, y)] \leq \widetilde{O}(\log n) \cdot d_G(x, y)$, 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 $d_T(u, v) = \sum_{e\in P_T(u,v)} \widehat{c}_T(e)$. *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 weights $c : E \to \mathbb{R}$ is called a* backboned graph *if there exists a spanning tree $T = (V, E_T)$ 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 [17, 1] probabilistically embed graphs into backboned graphs with small expected stretch.

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 A Small Collection of Covering Cycles

In this section, we show how we can augment connectivity for a demand pair s_i-t_i by showing that all its cuts can be *covered* by a small set of fundamental cycles (with respect to the base tree T) of low cost. Suppose G is a backboned graph that is an instance of the k-EC-ND problem with demand set

$\mathcal{D}$. 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-connects (where $l < k$) s_i-t_i for some $(s_i, t_i) \in \mathcal{D}$, and suppose H also contains the base tree path $P_T(s_i, t_i)$. The l-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 from H) which disconnects s_i and t_i 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 s_i-t_i l-cut 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. 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) *Take an instance of* k-EC-ND *with optimal solution* OPT. *Let $H \subseteq G$ l-connect s_i-t_i for $l < k$, and let 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 3.3 and 3.4 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. 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 $C(t_i)$ does not have a parent edge. 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, and C_1, C_4, C_5, C_{10} are the star components.

Since each edge in Q belongs to a distinct path in $\mathcal{P}_i$ (Q is a minimal l-cut separating s_i-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

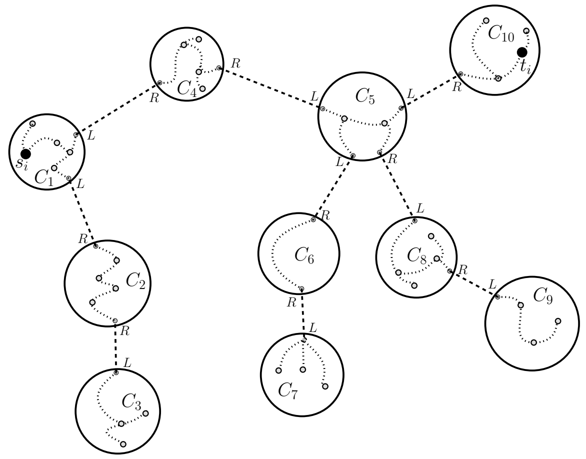


Figure 3.1. Example of the Portal Graph: circles are components, the dashed edges are portal edges, and dotted edges are other base-tree edges.

remainder of the proof, an edge $\{u, v\}$ which satisfies this property is said to *cover* the cut Q.

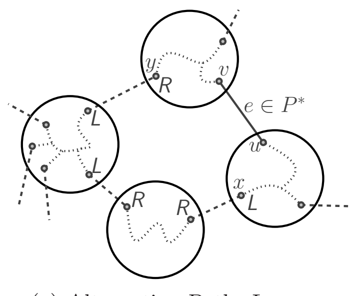
The Canonical Sequence: Since s_i and t_i can be k-connected in G, there must be a s_i-t_i path P^* *contained in the optimal* k-EC *subgraph* with $P^* \cap Q = \emptyset$. We first eliminate some "redundant" edges from P^* and show that among the other edges, there is one that covers Q. First remove 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.

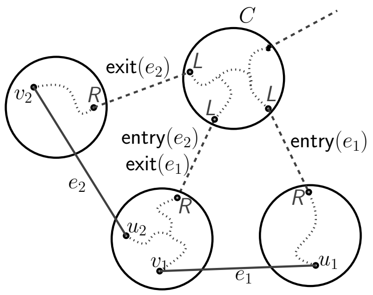
Lemma 3.4 (Alternating Paths Lemma) *Suppose none of the edges $\{u', v'\} \in P^*$ covers Q. For any edge $\{u, v\} \in P^*$, if the first portal edge on $P_T(u, v)$ is crossed with signature $(L \to R)$, then the final portal edge on $P_T(u, v)$ is crossed with signature $(R \to L)$. Also, the portal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

ment 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 C would connect the entry vertex labeled R and exit vertex labeled L in $O_{\{u,v\}} \setminus Q$, again giving a contradiction. ■

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 $(L \to R)$, $(R \to L)$, $\ldots$, $(L \to R)$.*

Proof. If we have two consecutive portal edges e'_j, 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 a cut. ■

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 is in 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 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. We prove the lemma if C is a canonical component; the proof for the other case is very similar. 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 exists because 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. By Lemma 3.4, $\mathsf{exit}(e'_1)$ must have its L-vertex in C as well. Furthermore, 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 simpler, since we do not have edges e_1, e_2.

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 above lemma is very similar to that for the previous one. Essentially, we don't have edges of the form e_2 and e''_j, since C_{t_i} is the final component on the path P^*. 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. Since $C(u_1) = C(s_i)$, we know that $\mathsf{head}(C(u_1))$'s L-vertex is in C_{u_1}. By the Alternating Paths Lemma 3.6, the final portal edge on $P_T(u_1, v_1)$ had 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. Hence, since $C(v_1) \neq C(t_i)$, then by the Consistency Lemma 3.6, we get that $\mathsf{head}(C(v_1))$'s L-vertex must be in $C(v_1)$. Because $C(u_{j+1}) = C(v_j)$ for all j, we have $\mathsf{head}(C(u_2))$'s L-vertex is in $C(u_2)$, and 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. ■

3.3 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-connected in H: we now identify *sets* and *elements* such that the Hitting Set problem exactly captures the problem of augmenting H to $(l+1)$-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 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-connected (with $l < k$) in H. 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 $(H \cup (\cup_{e \in F} O_e))$ is a network that $(l+1)$-connects every terminal pair s_i-t_i.*

As a warmup, 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-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.

3.4 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 $M_l := \binom{m}{l} 2^l$ 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 $O_{\{u,v\}} \setminus Q^l$ 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. [3] would be $O(\log N \log M) = O(k \log^2 m)$-competitive on this hitting set instance as well. Formally, the algorithm is given in the next page.

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

```
1: let $H \leftarrow \emptyset$.
2: set up the instance $\mathcal{I}$ of online hitting set.
3: for each terminal pair $(s_i, t_i)$ that arrives do
4:     let $H \leftarrow H \cup P_T(s_i, t_i)$
5:     for $l = 1$ to $k - 1$ do
6:         while $s_i$-$t_i$ not $l + 1$-connected in $H$ do
7:             find some violated $l$-cut $Q$ between $s_i$-$t_i$ in $H$
               and its labeling w.r.t. $H$
8:             feed $(Q$, labeling$)$ to online hitting set algorithm;
               let its output be $F \subseteq E$
9:             let $H \leftarrow H \cup (\cup_{e \in F} O_e)$
10:        end while
11:    end for
12: 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-connecting $\{s_i, t_i\}_{i \leq \tau}$. 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. COST SHARES FOR METRIC k-EC-ND

We now consider the k-connectivity network design problems with *metric* costs: when the underlying graph is complete, and the edge costs satisfy the triangle inequality. We first give an $O(1)$-approximation algorithm and then show that it is $O(1)$-strict, implying $O(1)$-approximations for metric stochastic and rent-or-buy versions. Though algorithms with better approximation ratios are known for metric k-EC-ND, we present one on which our cost shares are based.

4.1 An Algorithm for the Metric Case

Consider an instance $G = (V, E)$ of k-EC-ND with terminal pairs in $\mathcal{D}$; let D represents 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:

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

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 is based on ideas Cheriyan and Vetta [15] used for metric node connectivity problems: run the AKR algorithm ([2]) to 1-connect the demand pairs, then k-connect u and v (using neighbors) if AKR buys (u, v).

Algorithm 2 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 4.1 *The network S output by algorithm* MetricAlg *k-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. We first show that the 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. Further, we also observe that if $(u, v) \in F$, u and v are k-connected in S.

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 [15] for node-connectivity SNDP):

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

Let us explain why these are true: In any feasible solution, each terminal has to connect to at least k distinct neighbors. This coupled with the fact that each edge could be counted at most twice in the neighborhoods gives us the first bound.

The second bound follows from the fact that scaling a feasible LP solution to k-EC-ND by a factor k is feasible to the Steiner forest LP. We lose an additional factor 4 due to the approximation guarantee of the AKR algorithm, and because $\widetilde{F}$ is an Euler tour of F. The total cost of S is then

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

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. ■

4.2 Getting Strict Cost Shares

As basis for our cost sharing scheme, we use the cost shares associated with the AKR algorithm, as given by Fleisher et al. [21]. We refer to this cost sharing scheme as the FKLS analysis. In the following, let $F^{\mathcal{D}}$ denote the AKR solution on demand set $\mathcal{D}$. The FKLS analysis defines functions $\xi' : E\times V \to \mathbb{R}$ and $\xi : \mathcal{D}\to\mathbb{R}$ as follows. Each edge $e\in F^{\mathcal{D}}$ is assigned two *witness* terminals w_1 and w_2 such that $\xi'(e,w_1) = \xi'(e,w_2) = c_e/4$. $\xi'(e,v)$ is set to 0 for all $v\in V\setminus\{w_1,w_2\}$. For edges not in $F^{\mathcal{D}}$, $\xi'(e,u)=0$ for all u. The total cost share of a terminal pair (s_i,t_i) is then $\xi((s_i,t_i)) = \sum_{e\in F^{\mathcal{D}}} (\xi'(e,s_i)+\xi'(e,t_i))$. Also, for $e\in F^{\mathcal{D}}$, let τ_e denote the time at which e was bought by the AKR algorithm; denote the time at which (s_i,t_i) gets connected in $F^{\mathcal{D}}$ by τ_i. 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 $\tau_e\leq\tau_i$. 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. $\sum_{i=1}^{|\mathcal{D}|}\xi((s_i,t_i)) \leq \frac{2}{k}c(\mathsf{OPT}_{\mathsf{LP}}(\mathcal{D}))$. 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-connectivity problem has nice witness properties, the most natural thing to try would be to define cost shares for k-connectivity in the following way: for any edge e that s_i or t_i witness, the cost share for (s_i,t_i) includes the cost of k-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-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. Observing this, we 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.) This would help us get the desired cost shares.

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} $(=\emptyset$ 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}.

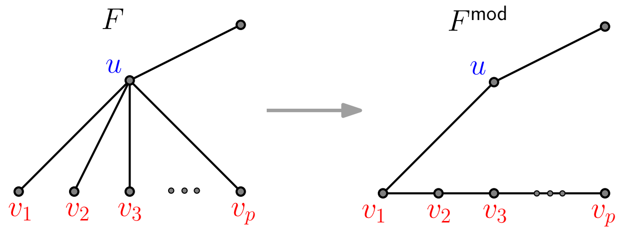


Figure 4.3. A step in the modification: u is the main vertex being altered

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 4.2 *The forest F^{mod} created by the above modification is such that (i) $c(F^{\mathsf{mod}})\leq 2c(F^{\mathcal{D}})$, (ii) vertices in F^{mod} have degree at most 3, and (iii) there exists witness definitions such that:*

- *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 the running* AKR *algorithm on demand set $\mathcal{D}\setminus(s_i,t_i)$.*
- *At most 2 terminals witness any edge in F^{mod}.*

Proof. Conditions (i) and (ii) follow directly from the construction of F^{mod}. We prove (iii) 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 u is being altered. There are two cases 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 main the 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.

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, then we know that each of the edges $(u, v_{r+1}), \ldots, (u, v_q)$ are witnessed by s_i or t_i - meaning 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 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-connect s_i and t_i.

Augmentation Algorithm: Augment $((s_i, t_i))$: For all $(u, v) \in W_i$, k-connect u and v using minimum cost.

Analysis: From Lemma 4.2, 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 $W_i \cup F^{\mathcal{D}\setminus(s_i,t_i)}$. Therefore u and v would be k-connected in $\mathsf{Augment}((s_i, t_i)) \cup \mathsf{MetricAlg}(\mathcal{D} \setminus (s_i, t_i))$: consider an 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-connect u' and v'. If it is in $F^{\mathcal{D}\setminus(s_i,t_i)}$, then from Theorem 4.1, $\mathsf{MetricAlg}(\mathcal{D} \setminus (s_i, t_i))$ would k-connect u' and v'. Hence, all edges on the $u - v$ path contained in $W_i \cup F^{\mathcal{D}\setminus(s_i,t_i)}$ 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, 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 completes the proof.

The following theorem therefore follows.

Theorem 4.3 *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 *problems.*

Acknowledgments. We thank Chandra Chekuri, Amit Kumar and Stefano Leonardi for useful discussions.

References

[1] I. Abraham, Y. Bartal, O. Neiman. Nearly tight low stretch spanning trees. In *48th FOCS*. 2008.

[2] A. Agrawal, P. Klein, 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).

[3] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, J. Naor. The online set cover problem. In *35th STOC*, 100–105. 2003.

[4] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, J. S. Naor. A general approach to online network optimization problems. In *15th SODA*, 577–586. 2004.

[5] B. Awerbuch, Y. Azar, Y. Bartal. On-line generalized Steiner problem. *Theoret. Comput. Sci.*, 324(2-3):313–324, 2004.

[6] N. Bansal, R. Khandekar, V. Nagarajan. Additive guarantees for degree bounded directed network design. In *STOC'08: Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, 769–778. 2008.

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

[8] P. Berman, C. Coulston. On-line algorithms for steiner tree problems (extended abstract). In *29th STOC*, 344–353. 1997.

[9] N. Buchbinder, J. S. Naor. Improved bounds for online routing and packing via a primal-dual approach. In *46th FOCS*, 293–304. 2006.

[10] T. Chakraborty, J. Chuzhoy, S. Khanna. Network design for vertex connectivity. In *STOC'08: Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, 167–176. 2008.

[11] Y. H. Chan, W. S. Fung, L. C. Lau, C. K. Yung. Degree bounded network design with metric costs. In *48th FOCS*. 2008.

[12] C. Chekuri, A. Gupta, N. Korula, R. Krishnaswamy, A. Kumar. Cost-sharing for subgraph k-edge-connectivity, 2008. Unpublished.

[13] C. Chekuri, N. Korula. Single-sink network design with vertex connectivity requirements. In *FST&TCS*. 2008.

[14] J. Cheriyan, S. Vempala, A. Vetta. An approximation algorithm for the minimum-cost k-vertex connected subgraph. *SIAM J. Comput.*, 32(4):1050–1055 (electronic), 2003.

[15] J. Cheriyan, A. Vetta. Approximation algorithms for network design with metric costs. In *37th STOC*, 167–175. 2005.

[16] J. Chuzhoy, S. Khanna. Algorithms for single-source vertex-connectivity. In *48th FOCS*. 2008.

[17] M. Elkin, Y. Emek, D. A. Spielman, S.-H. Teng. Lower-stretch spanning trees. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, 494–503. ACM Press, New York, NY, USA, 2005.

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

[19] J. Fakcharoenphol, S. Rao, K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. System Sci.*, 69(3):485–497, 2004.

[20] L. Fleischer, K. Jain, D. P. Williamson. Iterative rounding 2-approximation algorithms for minimum-cost vertex connectivity problems. *J. Comput. System Sci.*, 72(5):838–867, 2006.

[21] L. Fleischer, J. Könemann, S. Leonardi, G. Schäfer. Simple cost sharing schemes for multicommodity rent-or-buy and stochastic steiner tree. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, 663–670. ACM Press, New York, NY, USA, 2006.

[22] M. X. Goemans, A. V. Goldberg, S. Plotkin, D. B. Shmoys, Éva. Tardos, D. P. Williamson. Improved approximation algorithms for network design problems. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (Arlington, VA, 1994)*, 223–232. ACM, New York, 1994.

[23] M. X. Goemans, D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995. (Preliminary version in *5th SODA*, 1994).

[24] A. Gupta, A. Kumar, M. Pál, T. Roughgarden. Approximation via cost sharing: simpler and better approximation algorithms for network design. *J. ACM*, 54(3):Art. 11, 38 pp. (electronic), 2007.

[25] A. Gupta, M. Pál, R. Ravi, A. Sinha. Boosted sampling: Approximation algorithms for stochastic optimization problems. In *36th STOC*, 417–426. 2004.

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

[27] N. Immorlica, D. Karger, M. Minkoff, V. Mirrokni. On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems. In *15th SODA*, 684–693. 2004.

[28] K. 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).

[29] S. Khuller, B. Raghavachari. Improved approximation algorithms for uniform connectivity problems. *J. Algorithms*, 21(2):434–450, 1996.

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

[31] G. Kortsarz, R. Krauthgamer, J. R. Lee. Hardness of approximation for vertex-connectivity network design problems. *SIAM J. Comput.*, 33(3):704–720 (electronic), 2004.

[32] G. Kortsarz, Z. Nutov. Approximating node connectivity problems via set covers. *Algorithmica*, 37(2):75–92, 2003.

[33] L. C. Lau, J. Naor, M. R. Salavatipour, M. Singh. Survivable network design with degree or order constraints. In *39th STOC*, 651–660. 2007.

[34] L. C. Lau, M. Singh. Additive approximation for bounded degree survivable network design. In *STOC'08: Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, 759–768. 2008.

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

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

[37] D. P. Williamson, M. X. Goemans, M. Mihail, V. V. Vazirani. A primal-dual approximation algorithm for generalized Steiner network problems. *Combinatorica*, 15(3):435–454, 1995. (Preliminary version in *25th STOC*, 1993).

APPENDIX

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.

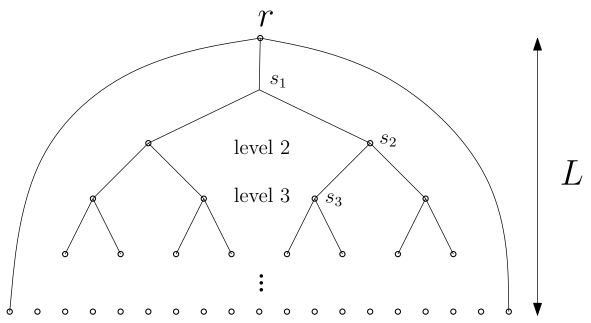


Figure A.4. A lower bound of $\Omega(L)$

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 descendent of the second request, and which is not yet connected to the root; in general, the next request is always chosen to be a descendent 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 descendent at level-$(2i-1)$ as the i-th request.