Streaming Algorithms for Network Design

Chandra Chekuri
Siebel School of Computing and Data Science, University of Illinois, Urbana, IL, USA

Rhea Jain
Siebel School of Computing and Data Science, University of Illinois, Urbana, IL, USA

Sepideh Mahabadi
Microsoft Research, Redmond, WA, USA

Ali Vakilian
Toyota Technological Institute at Chicago, IL, USA

Abstract

We consider the Survivable Network Design problem (SNDP) in the single-pass insertion-only streaming model. The input to SNDP is an edge-weighted graph $G = (V, E)$ and an integer connectivity requirement $r(uv)$ for each $u, v \in V$. The objective is to find a minimum-weight subgraph $H \subseteq G$ such that, for every pair of vertices $u, v \in V$, u and v are $r(uv)$-edge/vertex-connected. Recent work by [45] obtained approximation algorithms for edge-connectivity augmentation, and via that, also derived algorithms for edge-connectivity SNDP (EC-SNDP). In this work we consider *vertex-connectivity* setting (VC-SNDP) and obtain several results for it as well as improved results for EC-SNDP.

- We provide a general framework for solving connectivity problems including SNDP and others in streaming; this is based on a connection to *fault-tolerant spanners*. For VC-SNDP we provide an $O(tk)$-approximation in $\tilde{O}(k^{1-1/t}n^{1+1/t})$ space, where k is the maximum connectivity requirement, assuming an exact algorithm at the end of the stream. Using a refined LP-based analysis, we provide an $O(\beta t)$-approximation where β is the integrality gap of the natural cut-based LP relaxation. These are the first approximation algorithms in the streaming model for VC-SNDP. When applied to the EC-SNDP, our framework provides an $O(t)$-approximation in $\tilde{O}(k^{1/2-1/(2t)}n^{1+1/t} + kn)$ space, improving the $O(t \log k)$-approximation of [45] using $\tilde{O}(kn^{1+1/t})$ space; this also extends to element-connectivity SNDP.
- We consider vertex connectivity-augmentation in the link-arrival model. The input is a k-vertex-connected spanning subgraph G, and additional weighted links L arrive in the stream; the goal is to store the min-weight set of links such that $G \cup L$ is $(k+1)$-vertex-connected. We obtain constant-factor approximations in near-linear space for $k = 1, 2$. Our result for $k = 2$ is based on using the SPQR tree, a novel application for this well-known representation of 2-connected graphs.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Routing and network design problems

Keywords and phrases Streaming Algorithms, Survivable Network Design, Fault-Tolerant Spanners

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2025.4

Category APPROX

Related Version *Full Version*: `https://arxiv.org/abs/2503.00712` [17]

Funding *Chandra Chekuri*: Supported in part by NSF grant CCF-2402667.
Rhea Jain: Supported in part by NSF grant CCF-2402667.

Acknowledgements This work was conducted in part while Chandra Chekuri, Sepideh Mahabadi and Ali Vakilian were visitors at the Simons Institute for the Theory of Computing as part of the Data Structures and Optimization for Fast Algorithms program. The authors thank Ce Jin for his

contributions during the early stage of the project. The authors also thank an anonymous reviewer for pointers to efficient constructions and improved bounds for fault-tolerant spanners, and for pointing out an error in the previous version.

1 Introduction

Network design is a classical area of research in combinatorial optimization that has been instrumental in the development and advancement of several important algorithmic techniques. Moreover, it has practical applications across a wide variety of domains. In many modern real-world settings, graphs are extremely large, making it impractical to run algorithms that require access to the entire graph at once. This motivates the study of graph algorithms in the streaming model of computation; this is a commonly used model for handling large-scale or real-time data. There has been an extensive line of work on graph algorithms in the streaming setting. In addition to its practical utility, this line of work has led to a variety of new theoretical advances that have had auxiliary benefits well-beyond what was initially anticipated. Some well-studied problems in the streaming model include matching [56, 38, 6, 5, 46], max-cut [47], spanners [8, 30, 2, 50, 33, 34], sparsifiers [2, 50], shortest paths [32, 41], and minimum spanning tree [2, 66, 60], among many others.

We consider the Survivable Network Design problem (SNDP). The input to this problem is an undirected graph $G = (V, E)$ with non-negative edge-weights $w : E \to \mathbb{R}_{\geq 0}$ and a non-negative integer connectivity requirement $r(uv)$ for each unordered pair of vertices $u, v \in V$. The objective is to find a minimum-weight subgraph $H \subseteq G$ such that, for every pair of vertices $u, v \in V$, there exist $r(uv)$ disjoint uv-paths in H. If the paths for the pairs are required to be *edge-disjoint*, the problem is referred to as *edge-connectivity* SNDP (EC-SNDP), and if they are required to be *vertex-disjoint*, it is known as the *vertex-connectivity* SNDP (VC-SNDP). We refer to the maximum connectivity requirement as $k := \max_{uv} r(uv)$. SNDP is a fundamental problem that generalizes many well-known polynomial-time solvable problems, including minimum spanning tree (MST) and s-t shortest path, as well as several NP-hard problems, including Steiner Tree and Steiner Forest. We define some special cases of interest:

- *k-Connected Subgraph:* This is the special case in which $r(uv) = k$ for all $u, v \in V$. We denote the edge version as k-ECSS and the vertex version as k-VCSS; the goal is to find a min-weight k-edge-connected and k-vertex-connected spanning subgraph respectively.
- *Connectivity Augmentation:* In this problem, we are given a partial solution $G' = (V, E') \subseteq G$ for "free", with the guarantee that each $u, v \in V$ is at least $r(uv) - 1$ connected in G'. The goal is to find a min-weight set of edges $F \subseteq E \setminus E'$ that *augments* the connectivity of each u, v pair to $r(uv)$. The edges $E \setminus E'$ are often referred to as *links*. We refer to the augmentation version of the edge/vertex spanning problems as k-EC-CAP and k-VC-CAP respectively: here we are given a k-connected graph and the goal is to increase its connectivity to $k + 1$.

In this work, we study SNDP in the *insertion-only* streaming model. Formally, the algorithm reads the edges of a graph sequentially in an arbitrary order, processing each edge as it arrives. The goal is to solve graph problems over the streamed edges in a single pass, using a memory significantly smaller than storing the entire set of edges. While there has been significant recent progress for EC-SNDP in the streaming model, VC-SNDP remains essentially unexplored. One reason for this is that vertex-connectivity network design problems often do not share the same structural properties as their edge-connectivity counterparts. For instance, in the offline setting, EC-SNDP admits a 2-approximation via

a seminal iterated rounding algorithm of Jain [44]. On the other hand, the best known approximation for VC-SNDP is $O(k^3 \log n)$ [22]; moreover, the dependence on k is known to be necessary [16]. Another example highlighting the difficulty of vertex-connectivity problems is k-CAP: k-EC-CAP is known to reduce to 1 or 2-EC-CAP for all k [28] (see also [52, 18]), but no such structure exists for the vertex-connectivity setting. Thus the main motivating question for this paper is the following:

▶ **Question 1.** *What is the approximability of vertex-connectivity problems in the streaming setting?*

We briefly discuss prior work on streaming algorithms for EC-SNDP and several of its special cases. As mentioned earlier, streaming algorithms for shortest path and MST are both well-studied. We note an important distinction between these two problems: while MST admits an exact algorithm in $O(n)$ words of space, the current best known streaming algorithm for s-t shortest path is an $O(t)$ approximation in $\tilde{O}(n^{1+1/t})$-space. Thus, any problem that contains s-t shortest path as a special case incurs this limitation, while global connectivity problems such as MST are likely to have better trade-offs. Some other special cases of SNDP that have been studied in the streaming model include the Steiner forest problem in geometric setting [24], and testing k-connectivity [71, 72, 66, 4]. The k-EC-CAP problem was recently studied in the streaming model by [45], where they primarily considered two models:

- *Link arrival:* In this model, the partial solution G' is given up-front and does not count towards the space complexity of the algorithm. The weighted links $E \setminus E(G')$ arrive in the stream.
- *Fully streaming:* In this model, the edges of the partial solution, along with the additional weighted links used in augmentation, both arrive in the stream. They may arrive in an interleaved fashion; that is, the algorithm may not know the full partial solution when processing some links in $E \setminus E(G')$.

Both models are practically useful in their own right. Note that an α-approximation for k-CAP in the link-arrival model implies an (αk)-approximation for k-ECSS if we make k passes over the stream, as we can augment the connectivity of the graph by one in each pass. This is particularly useful in situations where k is a small constant and the best approximation ratio for link-arrival is significantly better than that of fully-streaming. In the link-arrival model, [45] obtained a tight $(2+\epsilon)$-approximation for k-EC-CAP in $O(\frac{n \log n}{\epsilon})$ space; note that while one can obtain a better than 2 in the offline setting, there is a lower bound of 2 in the semi-streaming setting. In the fully streaming model, they obtained an $O(t)$-approximation in $\tilde{O}(kn + n^{1+1/t})$ space for the connectivity augmentation problem using a *spanner* approach (a t-spanner is a sparse subgraph that preserves all distances to within a factor of t). By building on this, and using the reverse augmentation framework of Goemans *et al.* [39], they achieved an $O(t \log k)$-approximation for EC-SNDP with maximum connectivity requirement k; the space usage is $\tilde{O}(kn^{1+1/t})$. They further showed that any $O(t)$-approximation for this problem requires at least $\tilde{\Omega}(kn + n^{1+1/t})$ space.[1] Although the space complexity of their algorithm nearly matches the lower bound, their approximation for EC-SNDP is worse by a $\log k$ factor. A natural question is the following.

▶ **Question 2.** *Is there an $O(t)$-approximation for EC-SNDP using $\tilde{O}(n^{1+1/t} + kn)$-space?*

[1] While they only mentioned $\tilde{\Omega}(n^{1+1/t})$, it is straightforward to show that in general, the number of edges in an feasible solution of an SNDP instance with maximum connectivity requirement k can be as large as kn. Hence, kn is a trivial lower bound too.

1.1 Results and Techniques

We make significant progress towards our motivating questions and obtain a number of results; a summary is provided in Table 1. In all weighted graphs $G = (V, E)$ throughout the paper, we assume the weight function $w : E \to \{0, 1, \ldots, W\}$ where $W = n^{\text{polylog}(n)}$; thus $\log W = \text{poly}(\log n)$ and we do not explicitly include $\log W$ in the space complexity. We remark that designing streaming algorithms where the total number of stored edges is independent of W, as explored in [45], is an interesting technical question in its own right. We also note that all our algorithms assume an exact algorithm at the end of the stream, as the focus of this paper is optimizing the tradeoff between space and approximation ratio. We refer the reader to Section 2.1 for a discussion on how these algorithms can be made efficient.

A framework for SNDP. Our first contribution is a general and broadly applicable framework to obtain streaming algorithms with low space and good approximation ratios for SNDP; this is provided in Section 2. This framework applies to both edge and vertex connectivity, as well as an intermediate setting known as *element-connectivity*. [2] This framework addresses Question 1 and partially resolves Question 2 affirmatively. All the results below use $\tilde{O}(k^{1-1/t}n^{1+1/t})$-space for vertex-connectivity and element-connectivity problems, and $\tilde{O}(k^{1/2-1/(2t)}n^{1+1/t} + kn)$-space for edge-connectivity problems, where k is the maximum connectivity requirement and t is a parameter that allows for an approximation vs. space tradeoff. [3]

- For EC-SNDP, the framework yields $8t$-approximation, improving upon the $O(t \log k)$-approximation from [45]. For an $O(t)$-approximation, the space bound of our algorithm is off by only a factor of $k^{1/2-1/(2t)}$ from the known lower bound. These bounds also hold for element-connectivity, providing the first such results for this variant.
- For VC-SNDP, the framework yields an $O(tk)$-approximation.
- For VC-SNDP, the framework yields an $O(\beta t)$-approximation where β is the integrality gap of the natural LP relaxation. Using this, we obtain improved algorithms for several important special cases including k-VCSS and k-VC-CAP.

▶ Remark 1. Extending the lower bound construction from [45] to the vertex connectivity setting, we show that approximating VC-SNDP within a factor better than $2t + 1$ requires $\Omega(nk + n^{1+1/t})$ space, which points out that our upper bounds are nearly tight. The formal statement and proof of the lower bound is in Section 4.

Algorithms for k-VC-CAP

Our second contribution is for k-VC-CAP in the link-arrival model; this is provided in Section 3. This is a global connectivity problem and does not include the s-t-shortest path problem as a special case; hence we hope to obtain better bounds that avoid the overhead of using spanners. We obtain the following approximation ratios for k-VC-CAP; both algorithms use $\tilde{O}(n/\epsilon)$ space: (a) for $k = 1$, we obtain a $(3 + \epsilon)$-approximation algorithm, and (b) for $k = 2$, we obtain a $(7 + \epsilon)$-approximation algorithm. Following the earlier discussion, this implies that with k passes of the stream, we obtain a constant-factor approximation in near-linear space for k-VCSS for $k \leq 3$.

---

[2] Here, the vertex set V is divided into two types: *reliable* (R) and *non-reliable* ($V \setminus R$). Elements denote the set of edges E and the set of non-reliable vertices $V \setminus R$. For $u, v \in R$, a set of uv-paths are *element-disjoint* if they are disjoint on elements.

[3] When t is an even integer, the space used is a factor of $k^{1/(2t)}$ more than when t is odd. The results stated in this section assume that t is an odd integer.

Table 1 Summary of our results.

Problem	Approx.	Space	Note
EC-SNDP	$O(t \log k)$	$\tilde{O}(kn^{1+\frac{1}{t}})$ [45]	
	$8t$	$\tilde{O}(k^{\frac{1}{2}-\frac{1}{2t}}n^{1+\frac{1}{t}}+kn)$ [Here]	
	$O(t)$	$\Omega(kn+n^{1+\frac{1}{t}})$ bits [45]	lower bound
k-VC-CAP	$O(t)$	$\tilde{O}(k^{1-\frac{1}{t}}n^{1+\frac{1}{t}})$ [Here]	fully streaming ($n=\Omega(k^3)$)
	$O(t)$	$\Omega(n^{1+\frac{1}{t}})$ bits [Here]	fully streaming lower bound
	$3+\epsilon$	$\tilde{O}(n/\epsilon)$ [Here]	$k=1$ (link arrival)
	$7+\epsilon$	$\tilde{O}(n/\epsilon)$ [Here]	$k=2$ (link arrival)
VC-SNDP	$2tk$	$\tilde{O}(k^{1-\frac{1}{t}}n^{1+\frac{1}{t}})$ [Here]	
	$O(t)$	$\Omega(kn+n^{1+\frac{1}{t}})$ bits [Here]	lower bound
k-VCSS	2	$O(kn)$ [71, 19, 59]	for unweighted graphs
	$O(t)$	$\tilde{O}(k^{1-\frac{1}{t}}n^{1+\frac{1}{t}})$ [Here]	$n=\Omega(k^3)$
	$O(t)$	$\Omega(kn+n^{1+\frac{1}{t}})$ bits [Here]	lower bound

Techniques. Our general framework for SNDP and related problems is based on a connection to *fault-tolerant spanners* which were first introduced in [55] in geometric settings and then extensively studied in the graph setting. These objects generalize the notion of spanners to allow faults and are surprisingly powerful. Recent work has shown that, similar to ordinary spanners, optimal vertex fault-tolerant spanners and near-optimal edge fault-tolerant spanners can be constructed using a simple greedy algorithm, which enables their use in the streaming setting [12, 9]. Using these spanners alone suffices to obtain an $O(tk)$-approximation. Obtaining our refined approximation bounds requires additional insight: we combine the use of fault-tolerant spanners with an analysis via natural LP relaxations for SNDP. This improved analysis has a key benefit. It allows us to directly improve the factor $O(tk)$ for problems with corresponding integrality gap better than k – this applies to several of the problems we consider, such as EC-SNDP, ELC-SNDP and k-VCSS.

For 1-VC-CAP, we aim to augment a spanning tree to a 2-vertex-connected graph. To do so, we borrow ideas from the edge-connectivity setting: we root the tree arbitrarily, and for each node $u \in V$, we store the link that "covers" the most edges in the path from u to the root. This does not quite suffice in the vertex-connectivity setting, thus some additional care is required. Our main contribution is an algorithm for 2-VC-CAP. Here we need to augment a 2-vertex-connected graph to a 3-vertex-connected graph. In edge-connectivity, this reduces to cactus-augmentation (which can essentially be reduced to augmenting a cycle). However this does not hold for the vertex-connectivity setting. Instead, we use the SPQR-representation of a 2-vertex-connected graph: this is a compact tree-like data structure that captures all the 2-node cuts of a graph and was introduced by Di Battista and Tamassia [26] that dynamically maintains the triconnected components (and thus all 2-node cuts) of a graph. It was initially developed in the context of planar graphs [25, 27]. We use this tree-like structure to combine ideas from 1-VC-CAP with ideas from cactus augmentation in the edge-connectivity setting to obtain our result. The algorithm is tailored to the particulars of the SPQR representation and is technically quite involved. We refer the reader to Figure 4 for an example of a 2-connected graph and its SPQR tree.

1.2 Related Work

Offline Network Design. There is substantial literature on network design; here we briefly discuss some relevant literature. EC-SNDP admits a 2-approximation via the seminal work on iterated rounding by Jain [44] and this has been extended to ELC-SNDP [35, 21]. We note that the best known approximation for Steiner Forest, the special case with connectivity requirements in $\{0, 1\}$, is 2. Steiner tree is another important special case, and for this there is a $(\ln 4 + \epsilon)$-approximation [14]. Even for 2-ECSS the best known approximation is 2, although better bounds are known for the unweighted case. Recently there has been exciting progress on k-EC-CAP, starting with progress on the special case of weighted TAP (tree augmentation problem which corresponds to $k = 1$). For all k, k-EC-CAP admits a $(1.5 + \epsilon)$-approximation [69] – see also [67, 68, 13, 15, 37]. In contrast to the constant factor approximation results for edge-connectivity, the best known approximation for VC-SNDP is $O(k^3 \log n)$ due to Chuzhoy and Khanna [22]. Moreover, even for the single-source setting it is known that the approximation ratio needs to depend on k for sufficiently large k [16]. The single-source problem admits an $O(k^2)$-approximation [61, 63] and subset k-connectivity admits an $O(k \log^2 k)$-approximation [54]. Several special cases have better approximation bounds. When $k \leq 2$, VC-SNDP admits a 2-approximation [35, 21] which also implies that k-VCSS for $k = 2$ and k-VC-CAP for $k = 1$ admit a 2-approximation. For k-VCSS a $(4 + \epsilon)$-approximation is known when n is large compared to k [65, 20]. For smaller value of $k \leq 6$ improved bounds are known – see [64]. These are also the best known bounds for k-VC-CAP when n is large compared to k. For large k, k-VC-CAP and k-VCSS admit an $O(\log \frac{n}{n-k})$ and $O(\log k \log \frac{n}{n-k})$-approximation respectively with more precise bounds known in various regimes [64]. We note that many of the approximation results (but not all) are with respect to a natural cut-cover LP relaxation. This is important to our analysis since some of our results are based in exploiting the integrality gap of this LP relaxation. Many improved results are known for special cases of graphs including unweighted graphs, planar and graphs from proper minor-closed families, and graphs arising from geometric instances. We focus on general graphs in this work.

Streaming Graph Algorithms. Graph problems in the streaming model have been studied extensively, particularly in the context of compression methods that reduce the graph size and preserve connectivity within a factor of t (i.e., *spanners*) [31, 8, 30] or approximate cuts within a $(1+\epsilon)$ factor (i.e., *cut sparsifiers* and related problems) [1, 51, 48, 49]. These problems have also been widely explored in the *dynamic* streaming model, where edges are both inserted and deleted. While graph sketching approaches have sufficed to yield near-optimal algorithms for sparsifiers [2], the state-of-the-art for spanners in dynamic streams remained multi-pass algorithms until recently [2, 47]. Filtser *et al.* [34] developed the first single-pass algorithm for $\tilde{O}(n^{2/3})$-spanners using $\tilde{O}(n)$ space in dynamic streams. Though this result has a large approximation factor, Filtser *et al.* conjectured that it may represent an optimal trade-off.

Another line of research related to our work is testing connectivity in graph algorithms. This problem has been studied for both edge-connectivity and vertex-connectivity [31, 71, 66], as well as in dynamic settings [2, 23, 40, 7]. Specifically, k-connectivity for both edge- and vertex-connectivity can be tested using $\tilde{O}(nk)$ space even in dynamic streams [2, 7].

Finally, for the problem of k-ECSS, [45] designed a k-pass algorithm within the augmentation framework [39] that finds an $O(\log k)$-approximate solution using $\tilde{O}(nk)$ space.

Due to space constraints, this version of the paper serves as an extended abstract. Please see the full version [17] for more detailed description of technical details and related work.

1.3 Preliminaries

We provide preliminary background on connectivity. Let $G = (V, E)$ be an undirected graph. Two vertices $s, t \in V$ are k-edge (k-vertex) connected if G contains k edge-disjoint (vertex-disjoint) st-paths. Let $S \subset V$ be a subset of vertices. We denote the set of edges crossing S by $\delta_G(S)$; $\delta_G(S) = \{uv \in E(G) \mid u \in S, v \in V \setminus S\}$. We drop G when it is implicit from the context. Some additional notation (following prior work [61, 21, 35]) is required for vertex connectivity. A *biset* $\hat{S} = (S, S^+)$ is a pair of sets where $S \subseteq S^+ \subseteq V$. We say that an edge *crosses* a biset $\hat{S}$ if one of its endpoints is in S and the other is in $V \setminus S^+$. We define $\delta(\hat{S}) = \{uv \in E(G) \mid u \in S, v \in V \setminus S^+\}$.

Menger's theorem is a fundamental result characterizing the relation between the maximum connectivity of and the minimum cut between a pair of vertices. Its formulation is as follows:

▶ **Theorem 2** (Menger's theorem). *Two vertices $s, t \in V$ are k-edge connected iff for each set $S \subset V$ such that $s \in S$ and $t \in V \setminus S$, $|\delta(S)| \geq k$. Similarly, $s, t \in V$ are k-vertex connected iff for each biset $\hat{S} \subset V \times V$ such that $s \in S$ and $t \in V \setminus S^+$, $|\delta(\hat{S})| + |S^+ \setminus S| \geq k$.*

1.3.1 Fault-Tolerant Spanners in Streaming

▶ **Definition 3** (Fault-Tolerant Spanners). *A subgraph $H \subseteq G$ is an f-*vertex-fault-tolerant (f-VFT) *t-*spanner *of G if, for every subset of vertices $F_V \subseteq V$ of size at most f, all pairwise distances in $V \setminus F_V$ are preserved within a factor of t. That is, for all $u, v \in V \setminus F_V$, we have $d_{H \setminus F_V}(v, u) \leq t \cdot d_{G \setminus F_V}(v, u)$, where $G \setminus F_V$ and $H \setminus F_V$ denote the induced subgraphs $G[V \setminus F_V]$ and $H[V \setminus F_V]$, respectively. Similarly, we can define f-*edge-fault-tolerant (f-EFT) t-spanner.

A natural algorithm for constructing fault-tolerant spanners, which is a straightforward adaptation of the standard greedy algorithm of [3] for spanners, is to process the edges in an arbitrary order and add an edge (u, v) in the spanner if there exists a set of vertices of size at most f such that their removal increases the distance of u, v in the so-far-constructed spanner to at least $t + 1$.

▶ **Theorem 4** ([12]). *For any n-vertex graph G, the greedy algorithm for the f-VFT $(2t-1)$-spanner returns a feasible subgraph H of size $|E(H)| = O(f^{1-1/t} \cdot n^{1+1/t})$.*

This bound on spanner size is tight and fully matches the known lower bound for VFT [9].

▶ **Theorem 5** ([11]). *For any n-vertex graph G, the greedy algorithm for the f-EFT $(2t-1)$-spanner returns a feasible subgraph H of size*

$$|E(H)| = \begin{cases} O\left(t^2 f^{1/2-1/(2t)} n^{1+1/t} + tfn\right) & t \text{ is odd} \\ O\left(t^2 f^{1/2} n^{1+1/t} + tfn\right) & t \text{ is even.} \end{cases}$$

The bound on the spanner size is tight for constant odd t, and is off from the best known lower bound [9] by only quadratic factors of t for non-constant odd t, and only $t^2 f^{1/(2t)}$ for even t.

It is straightforward to show that for unweighted graphs, the greedy algorithm for spanners can be implemented in insertion-only streams with a space complexity equal to the size of H. Moreover, for the remainder of the paper, we assume that $t = O(\log n)$, as setting it any larger does not yield asymptotic improvements in the space complexity of the spanner.

Weighted graphs. Although running the greedy algorithm with edges in the increasing order of edge weights provides the same guarantee on size and stretch for fault-tolerant spanners (cf. [12]), the algorithm is no longer implementable in the streaming setting. A standard technique to address this issue is *bucketing*. Given a weighted graph $G = (V, E)$ with weight function $w : E \to \{0, 1, \cdots, W\}$, we partition the edges of G into $O(\epsilon^{-1} \log W)$ buckets, where the ith bucket contains edges with weights in the range $B_i := \left[(1+\epsilon)^{i-1}, (1+\epsilon)^i\right)$, for $i \geq 1$. We use bucket B_0 for zero-weight edges. Then, we construct a fault-tolerant spanner H_j in each bucket (treating it as an unweighted graph) using the greedy algorithm for unweighted f-FT spanners. This increases the space by a factor of $\epsilon^{-1} \log W$ and the spanner distortion by a factor of $(1+\epsilon)$. If $W = n^{\text{polylog}(n)}$ and by setting $\epsilon = 1/(2t-1)$, this process can be used to obtain a single-pass streaming algorithm that returns:

- An f-VFT $(2t)$-spanner of size $\tilde{O}(f^{1-1/t} \cdot n^{1+1/t})$ using $\tilde{O}(f^{1-1/t} \cdot n^{1+1/t})$ words of space;
- An f-EFT $(2t)$-spanner of size $\tilde{O}(f^{1/2-1/(2t)} \cdot n^{1+1/t} + fn)$ using $\tilde{O}(f^{1/2-1/(2t)} \cdot n^{1+1/t} + fn)$ words of space.

2 Generic Framework for Streaming Algorithms for Network Design

In this section, we describe our generic framework for streaming algorithms for network design problems. The algorithm, described in Algorithm 1, is both simple and practical.

Algorithm 1 A Generic framework for solving network design problems in streaming setting.

Input: Stream of weighted edges $\big(e, w(e)\big)$, network design problem $\mathcal{M}$ with maximum connectivity requirement k, approximation parameter t, and "offline" algorithm $\mathcal{A}$ for $\mathcal{M}$.
/* In Stream: */
construct an $O(kt)$-FT $(2t-1)$-spanner H of the edges with $\epsilon = \frac{1}{2t-1}$.
/* Postprocessing (*after stream terminates*): */
return the solution SOL of $\mathcal{M}$ on H output by the algorithm $\mathcal{A}$

As the algorithm only constructs and stores a fault-tolerant spanner with the given parameters $f = O(tk)$ and t in the stream, its space complexity follows directly from the discussion in Section 1.3.1. Specifically, the space complexity is $\tilde{O}\big(f^{1-1/t} \cdot n^{1+1/t}\big) = \tilde{O}\big(k^{1-1/t} \cdot n^{1+1/t}\big)$ in VC-SNDP and $\tilde{O}\big(f^{1/2-1/(2t)} \cdot n^{1+1/t} + fn\big) = \tilde{O}\big(k^{1/2-1/(2t)} \cdot n^{1+1/t} + kn\big)$ in EC-SNDP. However, the approximation *analysis* of the algorithms is technical. The rest of this section is dedicated to analyzing the approximation performance of Algorithm 1, and we will mainly focus on vertex-connectivity requirements. For this section, we assume the offline algorithm $\mathcal{A}$ used at the end of the stream is an exhaustive search: we enumerate all subgraphs of H and check feasibility for each, returning the solution of minimum cost. Note that this is possible since feasibility of SNDP (for both edge and vertex connectivity) can be checked in linear space by checking connectivity via augmenting path based max-flow routines. Section 2.1 contains a discussion of alternate choices of algorithm $\mathcal{A}$ and the resulting tradeoffs of runtime, space, and approximation ratio.

Before analyzing our framework, we present an observation on the structure of VFT spanners (or EFT spanners) that is used in our analysis for all variants.

▶ **Observation 6.** *Given a weighted graph $G = (V, E)$, let H denote the VFT spanner of G constructed by Algorithm 1 with parameters (t, f, ϵ). If $e \in E \setminus H_j$ with $w(e) \in B_j$, then H_j contains at least $L = \lfloor f/(t-1) \rfloor + 1$ vertex-disjoint uv-paths, each containing at most t edges that all have weights in B_j.*

Proof. We perform L iterations, and in each iteration i, we find a uv-path P_i of length at most t that is vertex-disjoint from the previously constructed uv-paths $P_1, \ldots, P_{i-1}$. To do this, we define the set of vertices $F_i = \big(P_1 \cup \cdots \cup P_{i-1}\big) \setminus \{u, v\}$ and find a uv-path in $H_j \setminus F_i$. Initially, set $F_1 = \emptyset$. Since $|F_i| \leq (t-1)(i-1) \leq (t-1)(L-1) \leq f$, by the properties of the f-VFT t-spanner H_j, there exists a uv-path $P_i \subseteq H_j \setminus F_i$, which is a path containing at most t edges, each with weight belonging to B_j, and is vertex-disjoint from $P_1, \ldots, P_{i-1}$. ◀

A Simple Analysis Based on Integral Solutions. Given any optimal solution OPT to the VC-SNDP instance on G, one can use Observation 6 to construct an $O(tk)$-approximate solution on the output H of Algorithm 1 as follows: for each $e = uv \in \text{OPT}$, if $e \in H$, add e to the solution. Else, add the vertex disjoint u-v paths given by Observation 6 to the solution. This gives us the following theorem; the formal proof is deferred to a full version [17].

▶ **Theorem 7.** *Let H be the VFT spanner of a weighted graph G as constructed in Algorithm 1 with parameters $(t, f = (2t-2)(k-1), \epsilon = 1/(2t-1))$. Then an optimal solution of VC-SNDP on (H, r) is within a $\big(2tk\big)$-factor of an optimal solution of VC-SNDP on (G, r).*

▶ **Corollary 8.** *There exists an algorithm for VC-SNDP in insertion-only streams that uses $\tilde{O}(k^{1-1/t} \cdot n^{1+1/t})$ space and outputs a $(2tk)$-approximate solution.*

An Improved Analysis via Fractional Solutions. We show a refined analysis of the framework which shows that an $O(tk)$-VFT $O(t)$-spanner contains an $O(t)$-approximate *fractional* solution for the given VC-SNDP instance. This is particularly interesting as it demonstrates that a fault-tolerant spanner preserves a near-optimal fractional solution for VC-SNDP on the given graph G. In other words, fault-tolerant spanners serve as *coresets* for network design problems. The standard (bi)cut-based relaxation is as follows:

$$\begin{aligned} \min \quad & \sum_{e \in E} w(e)x(e) & & \\ s.t. \quad & \sum_{e \in \delta(\hat{S})} x(e) \geq h(\hat{S}) & \forall S \subseteq S^+ \subseteq V & \qquad \text{(VC-SNDP-LP)} \\ & x(e) \geq 0 & \forall\, e \in E & \end{aligned}$$

Here, when $S \subseteq S^+$, $h(\hat{S})$ is defined as $\max(0, r(\hat{S}) - |S^+ \setminus S|)$, where $r(\hat{S}) := \max_{v \in S, u \in V \setminus S^+} r(u, v)$. Otherwise, $h(\hat{S}) = 0$. In particular, observe that for every biset $\hat{S}$ with $h(\hat{S}) > 0$, $h(\hat{S}) + |S^+ \setminus S| \leq k$.

▶ **Theorem 9.** *Let H be the VFT spanner of a weighted graph G as constructed in Algorithm 1 with parameters $(t, f = (2t-2)(2k-1), \epsilon = 1/(2t-1))$. Then the weight of an optimal* fractional *solution of VC-SNDP on (H, r) is within a $4t$-factor of the optimal solution of VC-SNDP on (G, r).*

We note a small but important difference between the algorithm implied by theorem above, and the one earlier. We increase the f by a factor of 2 which is needed for the proof below.

Proof. Let $H = H_1 \uplus \cdots \uplus H_T$, where $T = \epsilon^{-1} \log W$, be the output of Algorithm 1 with $f = (2t-2)(2k-1)$. Recall that for every $1 \leq j \leq T$, H_j is a $\big((2t-2)(2k-1)\big)$-VFT $(2t-1)$-spanner for the edges in G with weights in $B_j := ((1+\epsilon)^{j-1}, (1+\epsilon)^j]$.

Let OPT $\subseteq E$ be an optimal solution for the VC-SNDP instance on G. Starting from OPT, we construct a feasible fractional solution $\boldsymbol{x}$ supported only on H, for the standard (bi)cut-based relaxation (i.e., VC-SNDP-LP) of the VC-SNDP instance, with cost at most $O(t) \cdot w(\text{OPT})$.

Then, the fractional solution $\boldsymbol{x}$ is constructed from OPT as follows (in Figure 1). Note

Construction of $\boldsymbol{x}$ from OPT

Initialize $\boldsymbol{x}$ as an all-zero vector, i.e., $\boldsymbol{x}(e) = 0$ for all $e \in E$.
for each $e = (u, v) \in$ OPT
  let B_j denote the weight class to which the weight of e belongs.
  if $e \in H_j$ **then** $\boldsymbol{x}(e) \leftarrow 1$
  else let $P_1, \cdots, P_{2k}$ be vertex-disjoint uv-paths in H_j, each of length at most t
    for each $e' \in P_1 \cup \cdots \cup P_{2k}$, $\boldsymbol{x}(e') \leftarrow \min(1, \boldsymbol{x}(e') + 1/k)$

Figure 1 Construction of the fractional solution $\boldsymbol{x}$ of VC-SNDP-LP from OPT.

that our algorithm does not need to explicitly construct $\boldsymbol{x}$; this is only for analysis purposes.

First, we prove the feasibility of $\boldsymbol{x}$ for VC-SNDP-LP(G, w, h). Consider a biset $\hat{S}$ with $r(\hat{S}) > 0$. Note that for this to hold, it requires that $|S^+ \setminus S| < r(\hat{S})$. In the optimal solution OPT, there are at least $h(\hat{S})$ edges in $\delta_G(\hat{S})$. Let L_1 and L_0 respectively denote the number of edges in $\delta_{\text{OPT}}(\hat{S})$ that belong to H and those that do not; i.e., $L_1 = |\delta_{\text{OPT}}(\hat{S}) \cap H|$ and $L_0 = |\delta_{\text{OPT}}(\hat{S}) \setminus H|$. Note that since $\boldsymbol{x}(e) = 1$ for every $e \in H$, if $L_1 \geq h(\hat{S})$ then the fractional solution $\boldsymbol{x}$ satisfies the connectivity requirement of $\hat{S}$.

Next, consider the case in which $L_1 < h(\hat{S}) = k'$. We select $k' - L_1$ edges from the edge set $\delta_{\text{OPT}}(\hat{S}) \setminus H$ and denote them as $e_1 \coloneqq (u_1, v_1), \ldots, e_{k'-L_1} \coloneqq (u_{k'-L_1}, v_{k'-L_1})$. For each $i \leq k' - L_1$, consider the $2k$ vertex-disjoint (u_i, v_i)-paths $P_1, \ldots, P_{2k}$ in H_j, as shown in Observation 6, and used in the construction of $\boldsymbol{x}$ (see Figure 1). Since $P_1, \cdots, P_{2k}$ are vertex-disjoint and $u_i v_i$ is crossing the biset $\hat{S}$, at least

$$\begin{aligned} 2k - L_1 - |S^+ \setminus S| > 2k - L_1 - |S^+ \setminus S| - (k' - L_1) &= 2k - (k' + |S^+ \setminus S|) \\ &= 2k - (h(\hat{S}) + |S^+ \setminus S|) \geq k \end{aligned}$$

of the paths must have an edge crossing $\hat{S}$ distinct from $\delta_{\text{OPT}}(\hat{S}) \cap H$, where the last inequality holds because for every biset $\hat{S}$ with $h(\hat{S}) > 0$, $h(\hat{S}) + |S^+ \setminus S| \leq k$. Without loss of generality, let us denote k of these distinct edges by $e^i_1, \ldots, e^i_k$. Then, by the construction of $\boldsymbol{x}$,

$$\begin{aligned} \sum_{e \in \delta_H(\hat{S})} \boldsymbol{x}(e) \geq \sum_{e \in \delta_{\text{OPT}}(\hat{S}) \cap H} \boldsymbol{x}(e) + \sum_{e \in \{e^i_j\}_{i \in [k'-L_1], j \in [k]}} \boldsymbol{x}(e) &\geq L_1 + (k' - L_1) \cdot k \cdot \frac{1}{k} \\ &= k' = h(\hat{S}). \end{aligned}$$

Note that the second inequality holds because $\delta_{\text{OPT}}(\hat{S}) \cap H$ and $\{e^i_j\}_{i \in [k'-L_1], j \in [k]}$ are disjoint and no edge appears more than $k' - L_1$ times in the second summation (i.e., $\sum_{e \in \{e^i_j\}_{i \in [k'-L_1], j \in [k]}} \boldsymbol{x}(e)$). Hence, for every edge $e' \in \{e^i_j\}_{i \in [L], j \in [k]}$, its $\boldsymbol{x}$ value is at most $\min(1, c(e')/k) = c(e')/k$, because $c(e')$ which is defined as the number of times e' appears in the collection $\{e^i_j\}_{i \in [k'-L_1], j \in [k]}$, is at most $k' - L_1 \leq k' \leq k$. So, $\boldsymbol{x}$ is a feasible solution for VC-SNDP-LP(G, w, h).

For the cost analysis, note that for every edge $e = (u, v) \in \text{OPT}$ in weight class B_j, either $\boldsymbol{x}(e) = 1$, or it contributes in increasing the $\boldsymbol{x}$ values at most $2k \cdot (2t-1) \cdot \frac{1}{k} = 2 \cdot (2t-1)$ units on edges whose weight belong to B_j. Therefore, $w(\boldsymbol{x}) \leq 2 \cdot (2t-1) \cdot (1+\epsilon) \cdot w(\text{OPT}) = 4t \cdot w(\text{OPT})$. ◀

▶ **Corollary 10.** *There exists a $(4t\beta)$-approximation algorithm for VC-SNDP in insertion-only streams that uses $\tilde{O}(k^{1-1/t} \cdot n^{1+1/t})$ space, where β is the integrality gap of the cut-based LP relaxation.*

Implications for VC-SNDP and special cases. The main advantage of the fractional analysis is for those cases where the integrality gap of the LP is small. We point out some such cases and note that this is particularly useful for EC-SNDP and ELC-SNDP, which we discuss later.

- For k-VCSS there is an algorithm that uses $\tilde{O}(k^{1-1/t} \cdot n^{1+1/t})$ space and outputs a $(16+\epsilon)t$-approximate solution when n is sufficiently large compared to k. This follows from the known the integrality gap results for k-VCSS from [65, 20, 36]. This also implies a similar result for the connectivity augmentation problem k-VC-CAP.
- For finding the cheapest k vertex-disjoint s-t paths, there is an algorithm that uses $\tilde{O}(k^{1-1/t} \cdot n^{1+1/t})$ space and outputs a $4t$-approximate solution. This follows from the fact that the flow-LP is optimal for s-t disjoint paths.

We believe that the regime of k being small compared to n is the main interest in the streaming setting. For large values of k, $O(t \log(\frac{n}{n-k}))$ and $O(t \log k \cdot \log(\frac{n}{n-k}))$ approximation bounds can be derived for k-VC-CAP and k-VCSS, respectively, via known integrality gaps (see [64] and [62]). We omit formal statements in this version.

EC-SNDP. The proof technique that we outlined for VC-SNDP applies very broadly and also hold for EC-SNDP and ELC-SNDP. The proofs use the fact that the corresponding integrality gaps are 2 [44, 35]. We state below the theorem for EC-SNDP and defer ELC-SNDP and proof details to the full version [17].

▶ **Theorem 11.** *There exists a streaming algorithm for EC-SNDP that uses $\tilde{O}\big(k^{1/2-1/(2t)} \cdot n^{1+1/t} + kn\big)$ space and outputs an $(8t)$-approximate solution.*

2.1 Efficient Streaming Algorithms

The general framework and corresponding streaming algorithms discussed so far aim to optimize the tradeoff between space usage and approximation ratio. However, these algorithms, as described, may run in exponential time. In this section we outline approaches to obtain polynomial-time algorithms and the necessary space/approximation tradeoffs. We consider the two parts of the framework, (1) the streaming algorithm for storing a fault-tolerant spanner and (2) the algorithm at the end of the stream, separately.

Streaming Algorithm. The standard greedy algorithm for f-FT spanners runs in time $O(n^f)$ and this is not computationally efficient for large values of f. The study of polynomial-time constructions of such spanners is an active research question [29, 10, 11]. In particular, the following polynomial-time constructions can be implemented in the streaming setting. In all the methods described, the key modification is to replace the naïve $O(n^f)$-time (u, v) test (i.e., checking whether there exists a set of f edges or vertices whose removal increases the distance between u and v in the remaining subgraph to at least $2t$) with a polynomial-time (u, v) test.

- **VFT:** Bodwin, Dinitiz and Robelle [10] proposed a randomized implementation of the greedy approach for f-VFT $(2t-1)$-spanners. In their method, when testing whether to add an edge (u, v), the algorithm randomly samples $\Theta(\log n)$ induced subgraphs of the current spanner. Each subgraph includes u, v, and each of the remaining vertices is included independently with probability $1/(2f)$. Then, the edge (u, v) is added to the spanner if, in at least $1/4$ fraction of the sampled subgraphs, the distance between u and v exceeds $2t-1$. They show that this algorithm succeeds with high probability. It is simple to see that this can be implemented in the streaming setting in space that is proportional to the size of the spanner. Hence, it is possible to construct an f-VFT $(2t-1)$-spanner of optimal size, i.e., $\tilde{O}(f^{1-1/t}n^{1-1/t})$, in polynomial time in the streaming model, that succeeds with high probability.
- **EFT:** Dinitz and Robelle [29] provided a simple polynomial-time construction of f-EFT $(2t-1)$-spanners with slightly increased size complexity, i.e., $O(tf^{1-1/t}n^{1+t})$, which can be implemented efficiently in the streaming model too. Specifically, in their approach, when testing whether to add an edge (u, v), they perform f iterations as follows: starting with an initially empty set of edges F, in each iteration they attempt to find a path of length at most $(2t-1)$ between u and v in $S \setminus F$ and add it to F, where S is the current spanner. If such a path exists, it is added to F. If, in any iteration, no such path is found in $S \setminus F$, then the edge (u, v) is added to the spanner S. Otherwise, it is not added. Later, Bodwin, Dinitz, and Robelle [11] provided an improved analysis of the same (polynomial time) algorithm, achieving size bounds only slightly worse than those of the greedy approach with an exponential-time (u, v) test. Their result constructs f-EFT $(2t-1)$-spanners of size $O(t^{2.5}f^{1/2-1/(2t)}n^{1+1/t} + t^2fn)$ for odd t. For even t, the spanner size is $O(t^{2.5}f^{1/2}n^{1+1/t} + t^2fn)$.

Calculating Feasible Solution. In all the described algorithms, we perform an exponential time exhaustive search, by enumerating all possible solutions, on the constructed spanner to find an optimal solution after the stream terminates. The spanner effectively serves as a coreset for the problem. Since most of the problems considered in this paper are NP-hard, one could instead run known polynomial time approximation algorithms in the offline setting. We discuss some such approximation algorithms of interest.

The best known approximation for EC-SNDP is a 2-approximation via iterated rounding [44]. However, this requires exactly solving the cut-based LP; it is unclear if this can be done in linear space, especially given that this LP is exponentially sized. There is a primal-dual 2-algorithm for the *connectivity augmentation* problem [70]; this can be implemented in linear space. While the algorithm does not directly solve the LP, the analysis shows that it is a 2-approximation with respect to the optimal fractional solution. This can be repeatedly applied to obtain an $O(\log k)$-approximation (where k is the maximum connectivity requirement) for EC-SNDP [39]. For vertex-connectivity, The best known approximation for VC-SNDP employs a reduction to element connectivity. Given an α-approximation for ELC-SNDP that uses $f(m)$ space, one can obtain a $O(\alpha k^3 \log n)$-approximation in $O(m + f(m))$ space [22]. Some algorithms for special cases of VC-SNDP (such as single-source, k-VC-CAP, etc.) may be modified to run in linear space; we omit details for brevity and refer the reader to related work discussed in Section 1.2.

Combining the two parts, we achieve the following efficient streaming algorithms, where S is the space usage for the corresponding streaming problem (i.e. $S = k^{1-1/t} \cdot n^{1+1/t}$ for vertex-connectivity instances, and $S = k^{1/2-1/(2t)} \cdot n^{1+1/t} + kn$ for edge-connectivity instances):

- **Integral Solution Approach.** If there exists a polynomial time α-approximation algorithm for the SNDP problem with space complexity $g(m)$, where m is the size of the input graph, our framework returns an $O(\alpha tk)$-approximate solution in polynomial time, using $\tilde{O}(S + g(S))$ space.
- **Fractional Soultion Approach.** If there exists a polynomial time algorithm for the SNDP problem that returns an *integral* solution within an α_{fr}-factor of the optimal fractional solution to the cut-based LP relaxation of the problem, with space complexity $g(m)$ (where m is the size of the input graph), then our framework returns an $O(\alpha_{\text{fr}} \cdot t)$-approximate solution in polynomial time, using $\tilde{O}(S + g(S))$ space.

3 Vertex Connectivity Augmentation in Link-Arrival Model

We consider k-VC-CAP in the link arrival setting. Let $G = (V, E)$ denote the given underlying k-vertex-connected graph, and let $L \subseteq V \times V$, $w : L \to \mathbb{R}_{\geq 0}$ denote the weighted links arriving in the stream. Recall that we aim to find the min-weight subset $L' \subseteq L$ such that $(V, E \cup L')$ is $(k+1)$-vertex-connected. For ease of notation, we write k-connected to mean k-vertex-connected. We show that for $k = 1, 2$, we can obtain constant-factor approximations in near-linear space.

3.1 One-to-Two Augmentation

We outline the proof of the following theorem; details are deferred to the full version [17].

▶ **Theorem 12.** *There exists a streaming algorithm for 1-VC-CAP with edge weights $w : E \to [1, W]$, in an insertion-only stream, that uses $O(n\epsilon^{-1} \log W)$ space and outputs a $(3+\epsilon)$-approximate solution.*

We assume without loss of generality that the 1-connected graph G is a tree; if not, we can fix a spanning tree of G as the underlying graph, and consider all remaining edges as 0-weight links in the stream. For $i \geq 1$, we let B_i denote the set of links with $w(e) \in \left[(1+\epsilon)^{i-1}, (1+\epsilon)^i\right)$. We fix a root r of the tree G. For each $u \in V$, we let G_u denote the subtree of G rooted at u and $C(u)$ denote the set of children of u. For $u, v \in V$, we let $LCA(u, v)$ denote the lowest common ancestor of u and v in G; this is the vertex w furthest from the root such that $u, v \in G_w$. We overload notation and write $LCA(e)$ for $e \in E$ to be the LCA of its endpoints. We let $d_G(u, v)$ denote the number of edges in the unique tree path between u and v. We use the following lemma.

▶ **Lemma 13** ([57]). *Given any set of nodes V' with links $L \subseteq V' \times V'$ appearing in an insertion-only stream, one can store a minimum spanning tree on V' using $O(|V'|)$ memory space.*

The streaming algorithm is given in Algorithm 2. For each vertex $u \in V$ and each weight bucket j, we store the link $uv \in L \cap B_j$ with $LCA(u, v)$ closest to the root. Furthermore, for each $u \in G$, we consider a contracted graph with $C(u)$ nodes, where each subtree G_v for $v \in C(u)$ is contracted into a node. We maintain an MST on this contracted graph.

It is easy to see that the number of links stored is $O(n\epsilon^{-1} \log W)$. In order to bound the approximation ratio, we fix an optimal solution OPT on G. We will show that there exists a feasible solution SOL $\subseteq F$ such that $w(\text{SOL}) \leq (3+\epsilon)w(\text{OPT})$. We construct SOL as follows:

- For each $e = uv \in$ OPT, let j be such that $e \in B_j$. Add links $L_u(j)$ and $L_v(j)$ to SOL.
- For each $u \in V$:

Algorithm 2 The streaming algorithm for 1-to-2 VCSS augmentation.

Input: Weighted tree $G = (V, E)$ with root $r \in V$ and edge weights $w : E \to [1, W]$.
/* PREPROCESSING: */
for $u \in V$ **do**
  Initialize an empty dictionary L_u
  Construct $C'(u)$: for each $v \in C(u)$, contract G_v into one "supernode" and add it to $C'(u)$
  Define the graph T'_u with vertex set $C'(u)$ and edge set $\emptyset$
/* IN STREAM: */
for $e = uv \in L$ *in the stream* **do**
  Let j be such that $e \in B_j$
  if $L_u(j)$ *is undefined* ***or*** $L_u(j) = uv'$ *such that* $d_G(r, LCA(u, v)) < d_G(r, LCA(u, v'))$ **then**
    $L_u(j) \leftarrow e$
  if $L_v(j)$ *is undefined* ***or*** $L_v(j) = u'v$ *such that* $d_G(r, LCA(u, v)) < d_G(r, LCA(u', v))$ **then**
    $L_v(j) \leftarrow e$
  Update MST T'_x for $x = LCA(u, v)$ with link uv using Lemma 13
$F = \cup_{u \in V}(E(T'_u) \cup L_u)$
return An optimal solution on link set F

- For each $v \in C(u)$, if OPT contains a link between G_v and $G \setminus G_u$, mark v as "good";
- Contract all supernodes of $C'(u)$ corresponding to "good" vertices into one "good" supernode. Let $T''_u \subseteq T'_u$ be an MST on $C''(u)$. Add T''_u to SOL.

It is not difficult to verify that $w(\text{SOL}) \leq (3 + \epsilon)w(\text{OPT})$. We show feasibility in Lemma 14, concluding the proof of Theorem 12.

▶ **Lemma 14.** $(V, E \cup \text{SOL})$ *is a 2-connected graph.*

Proof. We want to show that for all $a \in V$, $(V, E \cup \text{SOL}) \setminus \{a\}$ is connected. Fix $a \in V$. Since OPT is feasible, it suffices to show that for all $uv \in \text{OPT}$ with $u, v \neq a$, there exists a u-v path in $E \cup \text{SOL}$ that does not use a. Fix $uv \in \text{OPT}$. If the (unique) u-v path in E does not contain a, then we are done. Thus we assume a is on the u-v tree path. We case on whether or not a is the LCA of u and v. Let $b = LCA(u, v)$. Let $u', v' \in C(b)$ be the children of b such that $u \in G_{u'}$ and $v \in G_{v'}$.

- **Case 1:** (see Fig 2) Suppose $a = b$. If u' or v' are not marked as "good", then u' and v' correspond to separate supernodes of $C'(a)$ and thus $G_{u'}$ and $G_{v'}$ must be connected via T''_a. $G_{u'}$ and $G_{v'}$ remain connected despite the deletion of a, so u and v are connected via T''_a.
  Suppose instead *both* u' and v' are marked as "good". Since $G_{u'}$ is good, there must be some link $e_u \in \text{OPT}[G_{u'}, G \setminus G_a]$. Let u'' be the vertex incident to e_u in $G_{u'}$, and let j be the weight class of e_u. Then $L_{u''}(j)$ must have an endpoint in $G \setminus G_a$, since $d_G(r, LCA(L_{u''}(j))) \leq d_G(r, LCA(e_u)) < d_G(r, a)$. Furthermore, $L_{u''}(j) \in \text{SOL}$, since $e_u \in \text{OPT}$. Similarly, there must be some $v'' \in G_{v'}$ such that $L_{v''} \cap \text{SOL}$ contains a link with one endpoint in $G \setminus G_a$. Since $G \setminus G_a$ remains connected despite the deletion of a, we obtain our desired u-v path.
- **Case 2:** (see Fig 3) If $a \neq b$, then a is either in $G_{u'}$ or in $G_{v'}$. Suppose without loss of generality that $a \in G_{u'}$; the other case is analogous. Let j be the weight class of uv. Then $d_G(r, LCA(L_u(j))) \leq d_G(r, b) < d_G(r, a)$; the first inequality holds since $v \in G \setminus G_b$.

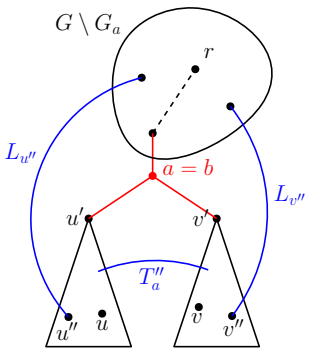


Figure 2 Example when $a = LCA(u, v)$. If u' or v' are not marked "good", u and v are connected via the link in T''_a shown. Else, they are connected using the links in $L_{u''}$ and in $L_{v''}$.

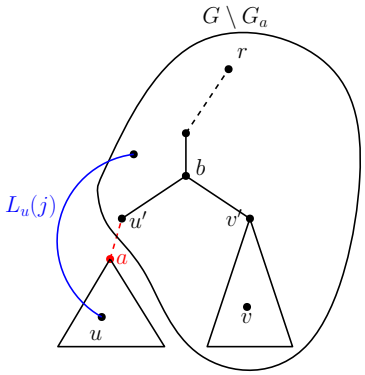


Figure 3 Example $a \neq LCA(u, v)$. Here $u \in G_a$, $v \in G \setminus G_a$, and $a \in G_{u'}$.

Thus $L_u(j)$ has one endpoint in $G \setminus G_a$. Furthermore, $L_u(j) \in$ SOL since $uv \in$ OPT. Since $G \setminus G_a$ remains connected despite the deletion of a, this gives us our desired u-v path in $E \cup$ SOL. ◀

3.2 Two-to-Three Augmentation

In this section, we outline the key ideas of the proof of the following theorem:

▶ **Theorem 15.** *There exists a streaming algorithm for 2-VC-CAP with edge weights $w : E \to [1, W]$, in an insertion-only stream, that uses $O(n\epsilon^{-1} \log W)$ space and outputs a $(7 + \epsilon)$-approximate solution.*

We provide background on SPQR trees in Section 3.2.1 and a technical overview of the 2-VC-CAP streaming algorithm in Section 3.2.2. The detailed algorithm and all proofs are deferred to the full version [17]. To avoid confusion with two-vertex cuts $\{u, v\}$, we denote edges/links as uv or (u, v). For any $E' \subseteq E \cup L$, $A, B \subseteq V$, we let $E'[A, B]$ denote the edges in E' with one endpoint in A and the other in B.

3.2.1 SPQR Trees

An SPQR tree is a data structure that gives a tree-like decomposition of a 2-connected graph into *triconnected* components. SPQR trees were first formally defined in [26] although they were implicit in prior work [25], and the ideas build heavily on work in [43]. Let $G = (V, E)$ be a 2-connected multigraph (parallel edges are allowed as they may arise during the recursive construction of SPQR trees).

▶ **Definition 16.** *A* separation pair *is a pair of vertices $a, b \in V$ such that at least one of the following hold:*

1. *$G \setminus \{a, b\}$ has at least two connected components $C_1, \ldots, C_k$. For each $i \in [k]$, we call the set $E[C_i] \cup E[C_i, \{a, b\}]$ a* separation class. *We let E' denote an additional separation class containing all parallel edges between a and b.*
2. *G contains at least two parallel edges ab and $G \setminus \{a, b\} \neq \emptyset$. Here the* separation classes *are $E_1 = \{e : e = ab\}$ (all parallel edges between a and b) and $E_2 = E(G) \setminus E_1$.*

Note that the separation classes partition the edge set $E(G)$.

▶ **Definition 17.** *For a separation pair $\{a, b\}$ with separation classes $E_1, \ldots, E_k$, choose $I \subseteq [k]$ such that both $E'_I = \cup_{i \in I} E_i$ and $E''_I = \cup_{i \notin I} E_i$ have at least 2 edges. The* split *operation introduces a new* virtual edge *$e = ab$ and replaces G with two new multigraphs $G' = (V(E'_I), E'_I \cup \{e\})$ and $G'' = (V(E''_I), E''_I \cup \{e\})$. Note that G' and G'' remain 2-connected.*

The SPQR tree T of a graph G is constructed via the following recursive procedure. Note that each node of T corresponds to a subgraph of G (with additional virtual edges).

1. If G has no separation pair, then T is the singleton tree with one node x and no edges. We write $G_x := G$ to denote the graph corresponding to tree node x.
2. Else, we split G on separation pair $\{a, b\}$ to obtain G', G'' and virtual edge $e = ab$. We recursively construct SPQR trees T', T'' on G', G''. Let x', x'' be the nodes of T', T'' respectively containing edge e – this is well defined, since the split operation partitions the edges of G and e is contained in both G' and G''. We let $T = T' \cup T'' \cup (x', x'')$; that is, we combine T' and T'' via an edge between x' and x''. We say the tree edge (x', x'') is associated with the virtual edge e.

If there are two adjacent tree nodes x, y such that G_x and G_y are both cycles *or* $|V(G_x)| = |V(G_y)| = 2$, we *merge* G_x and G_y by combining them and removing their shared virtual edge. After this process, for every $x \in V(T)$, G_x is either (1) exactly two vertices with three or more parallel edges, (2) a simple cycle, or (3) a three-connected graph with at least 4 vertices. These are referred to as P, S, and R nodes respectively. See Fig 4 for an example. SPQR trees can be constructed in linear-time [42].

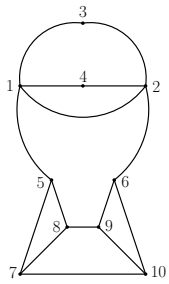
SPQR trees have the following nice properties, given mostly by [43, 26]. Each virtual edge is in exactly two tree nodes and is associated with a unique tree edge. Each edge in $E(G)$ is in exactly one tree node. The total size is bounded; that is, $\sum_{x \in V(T)} |E(G_x)| \leq 3|E(G)| - 6$. Most importantly, SPQR trees provide a nice characterization of all 2-cuts of G. Let $\{a, b\}$ be a 2-cut of G. Then, one of the following must be true:

1. $\{a, b\} = V(G_x)$ for a P-node x. The components of $G \setminus \{a, b\}$ correspond to subtrees of $T \setminus \{x\}$.
2. There exists a virtual edge $e = ab$ associated with a tree edge xy such that at least one of x and y is an R-node; the other is either an R-node or an S-node. The components of $G \setminus \{a, b\}$ correspond to subtrees of $(V(T), E(T) \setminus xy)$.
3. a and b are non-adjacent nodes of a cycle G_x for an S-node x. The components of $G \setminus \{a, b\}$ are the subtrees corresponding to the two components of $G_x \setminus \{a, b\}$.

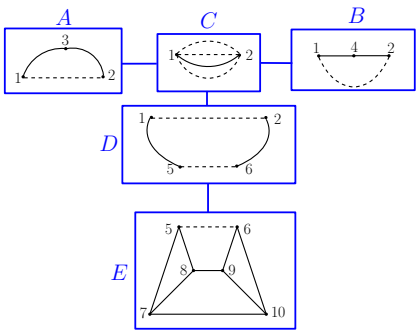
3.2.2 The Streaming Algorithm

We provide a high level description of the algorithm. Let $G = (V, E)$ be the given 2-connected graph, and let T be its SPQR tree. Let r be the root of T. There are three possible type of 2-vertex cuts $\{a, b\}$ in G. Intuitively, one can think of (1) as corresponding to a node being deleted in the tree, (2) corresponding to an edge being deleted in the tree, and (3) to handle connectivity within cycle nodes.

"Tree" Cuts. To handle 2-vertex-cuts of the first two types (see Figures 5, 6), we follow a similar approach to 1-VC-CAP (Section 3.1). Additional difficulty arises from the fact that T may contain multiple "copies" of each vertex in G; recall that if $u \in V$ is part of a separation pair, then it is duplicated during the "split" operation. Thus an incoming link uv does not necessarily correspond to a unique "tree link" $xy \in V(T) \times V(T)$. Our algorithm strategically chooses which tree link to assign each $uv \in L$ to in order to ensure feasibility.



(a) 2-connected graph G.



(b) SPQR Tree T with S-nodes A, B, D, P-node C, R-node E. Virtual edges are dashed.

Figure 4 Example of an SPQR tree T constructed from 2-connected graph G.

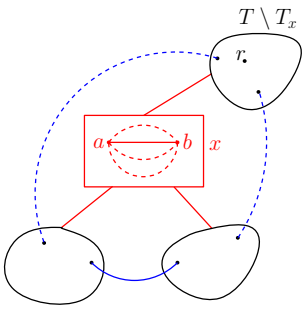


Figure 5 Example of 2-vtx-cut of type (1). The dashed blue links indicate links from subtrees to $T \setminus T_x$ (given by links stored to minimize LCA), while the solid blue link is between subtrees (given by MST).

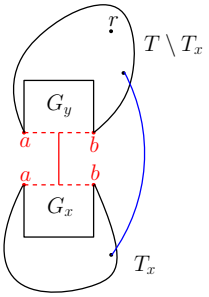


Figure 6 Example of 2-vtx-cut of type (2). The blue link indicates the type of link that must be included in any feasible solution.

For each tree node $x \in V(T)$, we store the link uv minimizing the distance from $LCA(x, y)$ to r, where x is the node containing u *closest* to the root and y is the node containing v *furthest* from the root. For each P-node x, we store an MST on the graph obtained by contracting all subtrees of x. The approximation and space analysis is similar to (although more technically involved than) that of 1-VC-CAP.

"Cycle" Cuts. We build on work by [52, 53, 45] for augmenting a cycle graph $C = \{1, \ldots, n\}$ to be 3-*edge*-connected in the *unweighted* setting. The idea is to bidirect links: each link uv is replaced with two *directed* links uv and vu. For a 2-edge cut $\{(i-1, i), (j, j+1)\}$, it suffices to include a directed link uv such that $u \notin [i, j]$ and $v \in [i, j]$. It is easy to see that for $1 \leq u \leq u' < v$, the directed link (u, v) covers a superset of the cuts covered by (u', v), and the same holds for for $v \leq u' < u \leq n$; see Fig 7. Thus we can store at most two incoming links per vertex and obtain a 2-approximation in linear space.

We employ a similar idea for covering 2-vertex cuts of type (3). Let x be some S-node. The main differences in this setting are handling vertex failures instead of edge failures, and, more importantly, the fact that two components of $G_x \setminus \{a, b\}$ can be connected via links not in G_x. For example, the components can be connected by a link between the two subtrees of T corresponding to the two components of $G_x \setminus \{a, b\}$. We handle this by first subdividing each virtual edge (by adding a dummy node), and then mapping each vertex in G to its

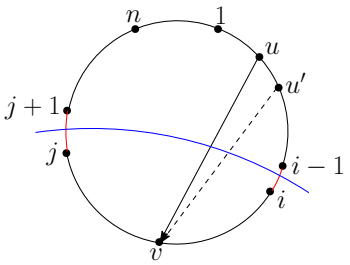


Figure 7 Example from [45] to augment a cycle to be 3-edge-connected. The blue curved line indicates the cut that needs to be covered.

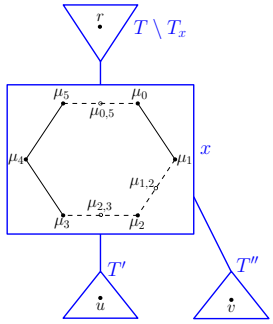


Figure 8 Example of S-node x with dummy vertices. Here u maps to $\mu_{2,3}$ and v maps to $\mu_{1,2}$.

corresponding point on the cycle (see Fig 8). We use this mapping to view each link $uv \in L$ as a link between two cycle nodes, and follow a similar approach to augmenting a cycle graph. The key technical challenge lies in the fact that a single link uv may map to multiple S-nodes in the tree, leading to a large blow-up in the approximation factor. We circumvent this by showing that for each $uv \in L$, we can restrict attention to at most three S-nodes in the tree: the ones containing u and v and the LCA of the corresponding tree nodes.

4 Lower Bounds for Streaming Network Design

We describe a lower bound for the vertex-connectivity tree-augmentation problem (VC-TAP), i.e., 1-VC-CAP.

▶ **Theorem 18.** *Consider the unweighted VC-TAP where both E and L arrives as a stream in an arbitrary order. Any single-pass algorithm returning a better than $(2t+1)$-approximation requires $\Omega(n^{1+1/t})$ space.*

Proof. Let $G = (V, E)$ be a fixed graph on n vertices with girth strictly larger than $2t+1$. Consider the INDEX problem: Alice has a bit string from $\{0,1\}^{E(G)}$, representing a subgraph $G' \subseteq G$. Alice then sends a message to Bob, who must recover the i-th bit of the string for a given index i, or equivalently, determine whether $(u,v) \in G'$ for a specified edge $(u,v) \in G$. It was shown by Miltersen et al. [58] that any bounded-error randomized protocol for INDEX requires a message of size $\Omega(|E(G)|)$ bits. Note that the graph G and its edges are known to both Alice and Bob.

We now use the streaming algorithm $\mathcal{A}$ for VC-TAP to design a protocol for the INDEX problem. Alice and Bob jointly construct a TAP instance (E, L) for $\mathcal{A}$ as follows:

- First, Alice sets $L := E(G')$ and feeds it into $\mathcal{A}$. She then sends the memory contents of $\mathcal{A}$ to Bob. To determine whether $(u,v) \in G'$, Bob constructs a chain $E := \{(x_1,x_2),(x_2,x_3),\ldots,(x_{|V|-1},x_{|V|})\}$ and feeds E to $\mathcal{A}$, where $x_1 := u$, $x_{|V|} := v$, and the intermediate vertices $\{x_2,\ldots,x_{|V|-1}\} = V \setminus \{u,v\}$ are ordered so that $d_H(u,x_{j-1}) \leq d_H(u,x_j)$ for $2 \leq j \leq |V|-1$. Here, the graph H is defined as G with the edge (u,v) removed.
- Bob then determines that $(u,v) \in G'$ if and only if $\mathcal{A}$ returns an approximate solution with cost less than $2t+1$ for the instance (E, L).

Next, we prove the correctness of the described protocol that uses $\mathcal{A}$.

- **If $(u,v) \in E(G')$:** In this case, the optimal solution for augmenting the chain E is to add the single edge $(u,v) \in L = E(G')$, completing a spanning cycle. Hence, $\mathcal{A}$ will report an approximate solution of cost at most $2t+1$ and Bob can correctly decide $(u,v) \in E(G')$.
- **If $(u,v) \notin E(G')$:** To show that Bob can correctly decide $(u,v) \notin E(G')$, it suffices to show that any feasible augmentation set $S = \{(x_{i_1}, x_{j_1}), (x_{i_2}, x_{j_2}), \ldots, (x_{i_s}, x_{j_s})\} \subseteq L = E(G')$ has size at least $2t+1$.
  We assume $i_k < j_k$ for all $1 \le k \le s$. Since $\{(x_1,x_2),(x_2,x_3),\ldots,(x_{|V|-1},x_{|V|})\} \cup S$ is 2-vertex-connected, the intervals $[i_1+1, j_1-1], \ldots, [i_s+1, j_s-1]$ covers all $\{2,\ldots,|V|-1\}$. This is because, once we add an edge (i_k, j_k), removing any vertex in $[i_k+1, j_k-1]$ does not disconnect the graph, as the connectivity is preserved by the edge (i_k, j_k). We can assume, without loss of generality, that $1 = i_1 < i_2 < \cdots < i_s$ and $j_1 < \cdots < j_s = |V|$, with $j_{k-1} > i_k$, by keeping a minimal feasible subset of S. This is because if $i_{k-1} < i_k$ and $j_{k-1} > j_k$, then removing (i_k, j_k) still leaves all cuts covered by (i_{k-1}, j_{k-1}). Moreover, if $j_{k-1} \le i_k$, the removing any nodes in $[j_{k-1}, i_k]$ will leave the graph $\{(x_1,x_2),(x_2,x_3),\ldots,(x_{|V|-1},x_{|V|})\} \cup S$ disconnected.
  Note that $S \subseteq E(G') \subseteq E(G) \setminus \{(u,v)\} = E(H)$. Now we inductively prove for every $1 \le k \le s$ that $d_H(u, x_{j_k}) \le k$. The base case $k = 1$ is immediate: $d_H(u, x_{j_1}) = d_H(x_{i_1}, x_{j_1}) = 1$. For the inductive step $2 \le k \le s$, we have

$$\begin{aligned} d_H(u, x_{j_k}) &\le d_H(u, x_{i_k}) + d_H(x_{i_k}, x_{j_k}) \\ &= d_H(u, x_{i_k}) + 1 \\ &\le d_H(u, x_{j_{k-1}}) + 1 && \triangleright \text{ by } i_k \le j_{k-1} < |V| \text{ and monotonicity of } d_H(u, x_*) \\ &\le k && \triangleright \text{ by induction hypothesis} \end{aligned}$$

  Hence, this shows that $d_H(u,v) = d_H(u, x_{j_s}) \le s$. So $G = H \cup \{(u,v)\}$ contains a cycle of length at most $s+1$. Since G has girth at least $2t+2$, we conclude $s \ge 2t+1$. ◀

Theorem 18 immediately implies the following for k-VC-CAP.

▶ **Corollary 19.** *Any algorithm approximating k-VC-CAP with a factor better than $2t+1$ in fully streaming requires $\Omega(n^{1+1/t})$ space.*

Moreover, by Theorem 18 and the fact that any feasible solution for k-VCSS and VC-SNDP (in general) has size $\Omega(nk)$, the following corollaries hold.

▶ **Corollary 20.** *Any algorithm approximating k-VCSS with a factor better than $2t+1$, in insertion-only streams, requires $\Omega(nk + n^{1+1/t})$ space.*

▶ **Corollary 21.** *Any algorithm approximating VC-SNDP with maximum connectivity requirement k with a factor better than $2t+1$, in insertion-only streams, requires $\Omega(nk + n^{1+1/t})$ space.*

References

1 Kook Jin Ahn and Sudipto Guha. Graph sparsification in the semi-streaming model. In *International Colloquium on Automata, Languages, and Programming*, pages 328–338. Springer, 2009. doi:10.1007/978-3-642-02930-1_27.

2 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Symposium on Principles of Database Systems (PODS)*, pages 5–14, 2012. doi:10.1145/2213556.2213560.

3 Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993. doi:10.1007/BF02189308.

4 Sepehr Assadi and Aditi Dudeja. A simple semi-streaming algorithm for global minimum cuts. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 172–180. SIAM, 2021. doi:10.1137/1.9781611976496.19.

5 Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In *Proceedings of the Symposium on Discrete Algorithms (SODA)*, pages 1723–1742, 2017. doi:10.1137/1.9781611974782.113.

6 Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 1345–1364, 2016. doi:10.1137/1.9781611974331.CH93.

7 Sepehr Assadi and Vihan Shah. Tight bounds for vertex connectivity in dynamic streams. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 213–227. SIAM, 2023. doi:10.1137/1.9781611977585.CH20.

8 Surender Baswana. Streaming algorithm for graph spanners–single pass and constant processing time per edge. *Inf. Process. Lett.*, 106(3):110–114, 2008. doi:10.1016/J.IPL.2007.11.001.

9 Greg Bodwin, Michael Dinitz, Merav Parter, and Virginia Vassilevska Williams. Optimal vertex fault tolerant spanners (for fixed stretch). In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1884–1900. SIAM, 2018. doi:10.1137/1.9781611975031.123.

10 Greg Bodwin, Michael Dinitz, and Caleb Robelle. Optimal vertex fault-tolerant spanners in polynomial time. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2924–2938. SIAM, 2021. doi:10.1137/1.9781611976465.174.

11 Greg Bodwin, Michael Dinitz, and Caleb Robelle. Partially optimal edge fault-tolerant spanners. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3272–3286. SIAM, 2022. doi:10.1137/1.9781611977073.129.

12 Greg Bodwin and Shyamal Patel. A trivial yet optimal solution to vertex fault tolerant spanners. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODS)*, pages 541–543, 2019. doi:10.1145/3293611.3331588.

13 Jarosław Byrka, Fabrizio Grandoni, and Afrouz Jabal Ameli. Breaching the 2-approximation barrier for connectivity augmentation: a reduction to steiner tree. In *Symposium on Theory of Computing (STOC)*, pages 815–825, 2020. doi:10.1145/3357713.3384301.

14 Jarosław Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *Journal of the ACM (JACM)*, 60(1):1–33, 2013. doi:10.1145/2432622.2432628.

15 Federica Cecchetto, Vera Traub, and Rico Zenklusen. Bridging the gap between tree and connectivity augmentation: unified and stronger approaches. In *Symposium on Theory of Computing (STOC)*, pages 370–383, 2021. doi:10.1145/3406325.3451086.

16 Tanmoy Chakraborty, Julia Chuzhoy, and Sanjeev Khanna. Network design for vertex connectivity. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 167–176, 2008. doi:10.1145/1374376.1374403.

17 Chandra Chekuri, Rhea Jain, Sepideh Mahabadi, and Ali Vakilian. Streaming algorithms for network design, 2025. doi:10.48550/arXiv.2503.00712.

18 Joseph Cheriyan, Tibor Jordán, and Ramamoorthi Ravi. On 2-coverings and 2-packings of laminar families. In *7th Annual European Symposium (ESA)*, pages 510–520. Springer, 1999.

19 Joseph Cheriyan, Ming-Yang Kao, and Ramakrishna Thurimella. Scan-first search and sparse certificates: an improved parallel algorithm for k-vertex connectivity. *SIAM Journal on Computing*, 22(1):157–174, 1993. doi:10.1137/0222013.

20 Joseph Cheriyan and László A Végh. Approximating minimum-cost k-node connected subgraphs via independence-free graphs. *SIAM Journal on Computing*, 43(4):1342–1362, 2014. doi:10.1137/120902847.

21 Joseph Cheriyan, Santosh Vempala, and Adrian Vetta. Network design via iterative rounding of setpair relaxations. *Combinatorica*, 26(3):255–275, 2006. doi:10.1007/S00493-006-0016-Z.

22 Julia Chuzhoy and Sanjeev Khanna. An $O(k^3 logn)$-approximation algorithm for vertex-connectivity survivable network design. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 437–441, 2009. doi:10.1109/FOCS.2009.38.

23 Michael S. Crouch, Andrew McGregor, and Daniel M. Stubbs. Dynamic graphs in the sliding-window model. In *Algorithms - ESA 2013 - 21st Annual European Symposium*, volume 8125 of *Lecture Notes in Computer Science*, pages 337–348. Springer, 2013. doi:10.1007/978-3-642-40450-4_29.

24 Artur Czumaj, Shaofeng H-C Jiang, Robert Krauthgamer, and Pavel Veselỳ. Streaming algorithms for geometric steiner forest. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, 2022.

25 Giuseppe Di Battista and Roberto Tamassia. Incremental planarity testing. In *30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 436–441, 1989.

26 Giuseppe Di Battista and Roberto Tamassia. On-line maintenance of triconnected components with spqr-trees. *Algorithmica*, 15(4):302–318, 1996. doi:10.1007/BF01961541.

27 Giuseppe Di Battista and Roberto Tamassia. On-line planarity testing. *SIAM Journal on Computing*, 25(5):956–997, 1996. doi:10.1137/S0097539794280736.

28 E A Dinits, Alexander V Karzanov, and Micael V Lomonosov. On the structure of a family of minimal weighted cuts in a graph. *Studies in Discrete Optimization*, pages 290–306, 1973.

29 Michael Dinitz and Caleb Robelle. Efficient and simple algorithms for fault-tolerant spanners. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pages 493–500, 2020. doi:10.1145/3382734.3405735.

30 Michael Elkin. Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners. *ACM Trans. Algorithms*, 7(2):20:1–20:17, 2011. doi:10.1145/1921659.1921666.

31 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2-3):207–216, 2005. doi:10.1016/J.TCS.2005.09.013.

32 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the data-stream model. *Journal on Computing*, 38(5):1709–1727, 2008. doi:10.1137/070683155.

33 Manuel Fernández V, David P Woodruff, and Taisuke Yasuda. Graph spanners in the message-passing model. In *Innovations in Theoretical Computer Science Conference*, 2020.

34 Arnold Filtser, Michael Kapralov, and Navid Nouri. Graph spanners by sketching in dynamic streams and the simultaneous communication model. In *Proceedings of the Symposium on Discrete Algorithms (SODA)*, pages 1894–1913, 2021. doi:10.1137/1.9781611976465.113.

35 Lisa Fleischer, Kamal Jain, and David P Williamson. Iterative rounding 2-approximation algorithms for minimum-cost vertex connectivity problems. *Journal of Computer and System Sciences*, 72(5):838–867, 2006. doi:10.1016/J.JCSS.2005.05.006.

36 Takuro Fukunaga, Zeev Nutov, and R Ravi. Iterative rounding approximation algorithms for degree-bounded node-connectivity network design. *SIAM Journal on Computing*, 44(5):1202–1229, 2015. doi:10.1137/13094503X.

37 Mohit Garg, Fabrizio Grandoni, and Afrouz Jabal Ameli. Improved approximation for two-edge-connectivity. In *Symposium on Discrete Algorithms (SODA)*, pages 2368–2410, 2023. doi:10.1137/1.9781611977554.CH92.

38 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Symposium on Discrete Algorithms (SODA)*, pages 468–485, 2012. doi:10.1137/1.9781611973099.41.

39 Michel X. Goemans, Andrew V. Goldberg, Serge A. Plotkin, David B. Shmoys, Éva Tardos, and David P. Williamson. Improved approximation algorithms for network design problems. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 223–232. ACM/SIAM, 1994. URL: http://dl.acm.org/citation.cfm?id=314464.314497.

40 Sudipto Guha, Andrew McGregor, and David Tench. Vertex and hyperedge connectivity in dynamic graph streams. In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 241–247, 2015. doi:10.1145/2745754.2745763.

41 Venkatesan Guruswami and Krzysztof Onak. Superlinear lower bounds for multipass graph processing. *Algorithmica*, 76:654–683, 2016. doi:10.1007/S00453-016-0138-7.

42 Carsten Gutwenger and Petra Mutzel. A linear time implementation of spqr-trees. In *International Symposium on Graph Drawing*, pages 77–90. Springer, 2000. doi:10.1007/3-540-44541-2_8.

43 J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–158, 1973. doi:10.1137/0202012.

44 Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21:39–60, 2001. doi:10.1007/S004930170004.

45 Ce Jin, Michael Kapralov, Sepideh Mahabadi, and Ali Vakilian. Streaming algorithms for connectivity augmentation. In *51st International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 297 of *LIPIcs*, pages 93:1–93:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ICALP.2024.93.

46 Michael Kapralov. Space lower bounds for approximating maximum matching in the edge arrival model. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1874–1893, 2021. doi:10.1137/1.9781611976465.112.

47 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Streaming lower bounds for approximating max-cut. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1263–1282. SIAM, 2014.

48 Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. *SIAM J. Comput.*, 46(1):456–477, 2017. doi:10.1137/141002281.

49 Michael Kapralov, Aida Mousavifar, Cameron Musco, Christopher Musco, Navid Nouri, Aaron Sidford, and Jakab Tardos. Fast and space efficient spectral sparsification in dynamic streams. In *Proceedings of the Symposium on Discrete Algorithms (SODA)*, pages 1814–1833, 2020. doi:10.1137/1.9781611975994.111.

50 Michael Kapralov and David Woodruff. Spanners and sparsifiers in dynamic streams. In *Proceedings of the Symposium on Principles of Distributed Computing*, pages 272–281, 2014. doi:10.1145/2611462.2611497.

51 Jonathan A Kelner and Alex Levin. Spectral sparsification in the semi-streaming setting. *Theory of Computing Systems*, 53(2):243–262, 2013. doi:10.1007/S00224-012-9396-1.

52 Samir Khuller and Ramakrishna Thurimella. Approximation algorithms for graph augmentation. *Journal of Algorithms*, 14(2):214–225, 1993. doi:10.1006/JAGM.1993.1010.

53 Samir Khuller and Uzi Vishkin. Biconnectivity approximations and graph carvings. *J. ACM*, 41(2):214–235, 1994. doi:10.1145/174652.174654.

54 Bundit Laekhanukit. An improved approximation algorithm for the minimum cost subset k-connected subgraph problem. *Algorithmica*, 72(3):714–733, 2015. doi:10.1007/S00453-014-9869-5.

55 Christos Levcopoulos, Giri Narasimhan, and Michiel Smid. Efficient algorithms for constructing fault-tolerant geometric spanners. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 186–195, 1998. doi:10.1145/276698.276734.

56 Andrew McGregor. Finding graph matchings in data streams. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 170–181. Springer, 2005. doi:10.1007/11538462_15.

57 Andrew McGregor. Graph stream algorithms: a survey. *ACM SIGMOD Record*, 43(1):9–20, 2014. doi:10.1145/2627692.2627694.

58 Peter Bro Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. On data structures and asymmetric communication complexity. *J. Comput. Syst. Sci.*, 57(1):37–49, 1998. doi:10.1006/JCSS.1998.1577.

59 Hiroshi Nagamochi and Toshihide Ibaraki. Linear time algorithms for finding k-edge connected and k-node connected spanning subgraphs. *Algorithmica*, 7(1992):583–596, 1992.

60 Jelani Nelson and Huacheng Yu. Optimal lower bounds for distributed and streaming spanning forest computation. In *Proceedings of the Symposium on Discrete Algorithms (SODA)*, pages 1844–1860, 2019. doi:10.1137/1.9781611975482.111.

61 Z. Nutov. Approximating minimum-cost connectivity problems via uncrossable bifamilies. *ACM Transactions on Algorithms (TALG)*, 9(1):1, 2012.

62 Zeev Nutov. Approximating minimum-cost edge-covers of crossing biset-families. *Combinatorica*, 34(1):95–114, 2014. doi:10.1007/S00493-014-2773-4.

63 Zeev Nutov. Erratum: Approximating minimum-cost connectivity problems via uncrossable bifamilies. *ACM Transactions on Algorithms (TALG)*, 14(3):1–8, 2018. doi:10.1145/3186991.

64 Zeev Nutov. Improved approximation algorithms for minimum cost node-connectivity augmentation problems. *Theory of Computing Systems*, 62:510–532, 2018. doi:10.1007/S00224-017-9786-5.

65 Zeev Nutov. A $4+\varepsilon$ approximation for k-connected subgraphs. *Journal of Computer and System Sciences*, 123:64–75, 2022.

66 Xiaoming Sun and David P Woodruff. Tight bounds for graph problems in insertion streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015.

67 Vera Traub and Rico Zenklusen. A better-than-2 approximation for weighted tree augmentation. In *Foundations of Computer Science (FOCS)*, pages 1–12, 2022.

68 Vera Traub and Rico Zenklusen. Local search for weighted tree augmentation and steiner tree. In *Symposium on Discrete Algorithms (SODA)*, pages 3253–3272, 2022. doi:10.1137/1.9781611977073.128.

69 Vera Traub and Rico Zenklusen. A (1.5+ ε)-approximation algorithm for weighted connectivity augmentation. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1820–1833, 2023. doi:10.1145/3564246.3585122.

70 David P. Williamson, Michel X. Goemans, Milena Mihail, and Vijay V. Vazirani. A primal-dual approximation algorithm for generalized steiner network problems. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, pages 708–717, New York, NY, USA, 1993. Association for Computing Machinery. doi:10.1145/167088.167268.

71 Mariano Zelke. k-connectivity in the semi-streaming model. *arXiv preprint cs/0608066*, 2006.

72 Mariano Zelke. Intractability of min-and max-cut in streaming graphs. *Information Processing Letters*, 111(3):145–150, 2011. doi:10.1016/J.IPL.2010.10.017.