Streaming Algorithms for Vertex Connectivity Network Design

Rhea Jain

University of Illinois at Urbana-Champaign

Joint work with Chandra Chekuri (UIUC), Sepideh Mahabadi (MSR Redmond), Ali Vakilian (TTIC)

Vertex Connectivity Network Design Problems

VC Survivable Network Design (VC-SNDP):

- **Input:** Graph $G=(V,E)$ with edge weights $w(e) \in \{0,\dots,W\}$; demand pairs $\{s_i, t_i\}$ with connectivity requirement k_i
- **Objective:** Min-weight subgraph $H \subseteq G$ that contains k_i-vertex disjoint s_i-t_i paths; i.e. s_i-t_i are k_i-connected
- **Notation:** $n=|V|, m=|E|, k=\max_i k_i$

VC Connectivity Augmentation (k-VC-CAP):

- Special case of VC-SNDP
- **Input:** "Partial solution" $(k-1)$-connected graph $G=(V,E)$; additional *links* L with weights $w(e) \in \{0,\dots,W\}$
- **Objective:** Min-weight set of links $L' \subseteq L$ such that $(V, E \cup L')$ is k-connected

Streaming Network Design

Edges arrive one at a time in stream. Two models for augmentation:

- **Fully Streaming:** E and L both arrive in stream in arbitrary order (may be interleaved)
- **Link Arrival:** E is given up-front, L arrives in stream

Goal: solve problem storing sublinear (in m) number of edges

Results

Existing results: Mostly in edge-connectivity (EC) setting [1]:

- EC-SNDP: $O(t\log k)$-approx in $\tilde{O}(kn^{1+1/t})$ space
- k-EC-CAP (both results with matching lower bounds):
  - Link Arrival: $(2+\varepsilon)$-approx in $\tilde{O}(n/\varepsilon)$ space
  - Fully Streaming: $O(t)$-approx in $\tilde{O}(nk+n^{1+1/t})$ space

Our results [2]

- VC-SNDP: $O(tk)$-approx in $\tilde{O}(k^{1-1/t}n^{1+1/t})$ space
- k-VC-CAP:
  - Link Arrival: $O(1)$-approx in $\tilde{O}(n)$ space for $k=1,2$
  - Fully Streaming: $O(t)$-approx in $\tilde{O}(k^{1-1/t}n^{1+1/t})$ for small k, $O(t\log\frac{n}{n-k})$-approx for large k

Note: several results in [2] not included in this poster, including improved bounds for EC-SNDP.

General Framework using Fault-Tolerant Spanners

Fault-tolerant Spanner: H is f-fault-tolerant t-spanner of G if for all vertex subsets $F \subseteq V$ with $|F| \le f$, for all $u,v \in V \setminus F$, $d_{H\setminus F}(u,v) \le t \cdot d_{G\setminus F}(u,v)$

Theorem [3]: In *unweighted* graphs, there is greedy algorithm for the f-VFT $(2t-1)$-spanner that returns subgraph of size $O(f^{1-1/t}\cdot n^{1+1/t})$.

- Greedy algorithm can be implemented in streaming!
- Use bucketing idea for weights: keep kt-fault-tolerant t-spanner H_j for each "bucket" of edges with weights between $[2^j, 2^{j+1}]$ up to $j=\log W$.
- Return $H = \cup_j H_j$

Our contribution: H contains a good approximate solution to VC-SNDP.

- Fix optimal solution OPT and use it to construct feasible solution in H
- **Key Observation:** In each spanner H_j, If $e=(u,v)$ is in bucket j and e is not in H_j, then H_j contains k vertex-disjoint u-v paths of length at most t

"Integral" Analysis:

- For each $e=(u,v) \in$ OPT, if $e \in H$, include in solution. If not, include k disjoint u-v paths from H_j.
- Total weight = $O(kt) \cdot w(\text{OPT})$

"Fractional" Analysis:

Idea: Construct *fractional* solution to natural LP instead of integral solution

- Variable $x_e \in [0,1]$ for each $e \in G$
- Constraint for each "vertex cut": for each disjoint $S, C \subseteq V$ with $|C| \le k-1$, must be at least one unit of flow from S to $V-(S\cup C)$.

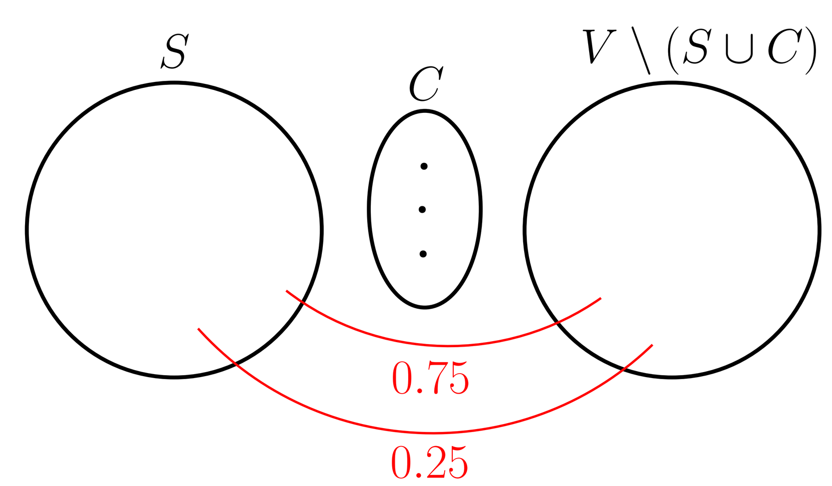


Figure 1. Example vertex cut with feasible fractional solution.

Note: Use $2kt$ as fault-tolerance instead of kt

- For each $e=(u,v) \in$ OPT, if $e \in H$, set $x_e = 1$. If not, add $\frac{1}{k}$ to $x_{e'}$ for each e' on the $2kt$ disjoint u-v paths.
- Total *fractional* weight = $O(t)$
- There exists integral solution of weight $O(\beta t)$; where β is integrality gap

Integrality gap for k-VC-CAP is $O(1)$ for $k=O(n^{1/3})$ and $O(\log\frac{n}{n-k})$ for large k.

Link-Arrival 1-to-2 Augmentation

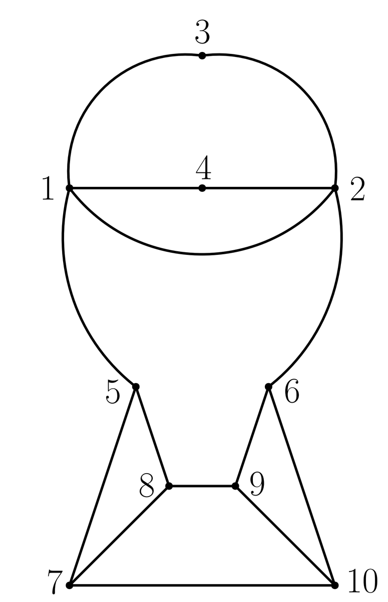
Objective: Augment a tree with additional links to be 2-connected. Assume tree is rooted

- For each node, store link with LCA closest to root for each weight class
- For each node, store MST on child subtrees

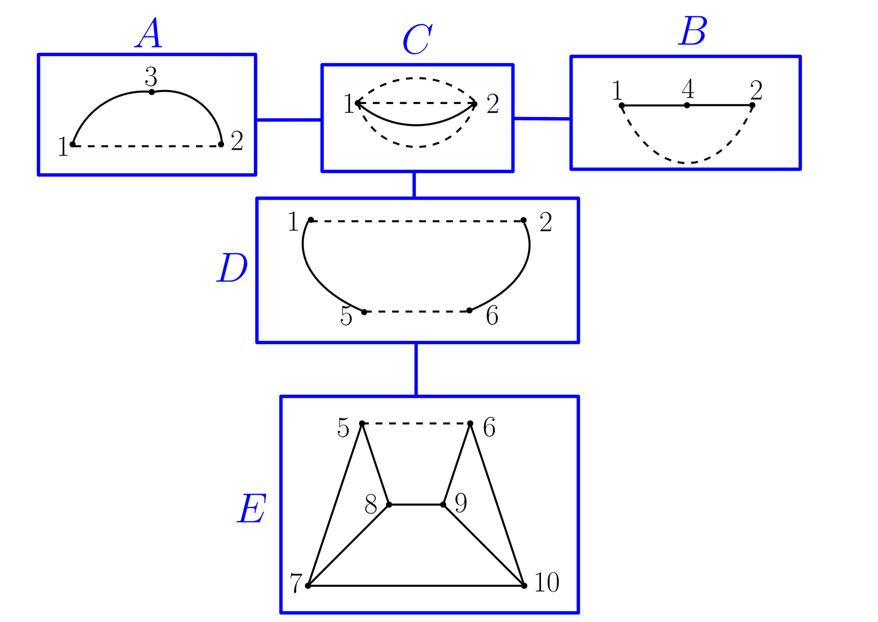
Total space: $O(n)$; Approx ratio: $3+\varepsilon$

Link-Arrival 2-to-3 Augmentation

Main idea: Use SPQR tree: data structure decomposing 2-connected graph into 3-connected components (see Figure 2b). Each "node" in tree is cycle (S-node), dipole (P-node) or 3-connected (R-node).



(a) 2-connected graph



(b) SPQR Tree: S-nodes A, B, D, P-node C, R-node E

All 2-node cuts correspond to one of the following:

1. P-node: use ideas from 1-to-2 Augmentation
2. Edge on SPQR tree: use ideas from 1-to-2 Augmentation
3. Non-adjacent vertices in S-node: use ideas from 2-to-3 *edge* connectivity augmentation

Total space: $O(n)$, Approx ratio: $7+\varepsilon$

References

[1] 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.

[2] Chandra Chekuri, Rhea Jain, Sepideh Mahabadi, and Ali Vakilian. Streaming algorithms for network design, 2025.

[3] 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.