Streaming Algorithms for Connectivity Augmentation Problems

Ali Vakilian

TOYOTA TECHNOLOGICAL INSTITUTE AT CHICAGO

Joint work with Ce Jin (MIT), Michael Kapralov (EPFL) and Sepideh Mahabadi (MSR)

Problem Statement

Connectivity Augmentation Problem (k-CAP)

Input:

- $(k-1)$-edge-connected graph $G = (V, E)$, and
- set of weighted links L, where weights are in $\{0,1, \dots, W\}$

Output: min-weight $L' \subset L$ s.t. $G' = (V, E \cup L')$ is k-edge connected

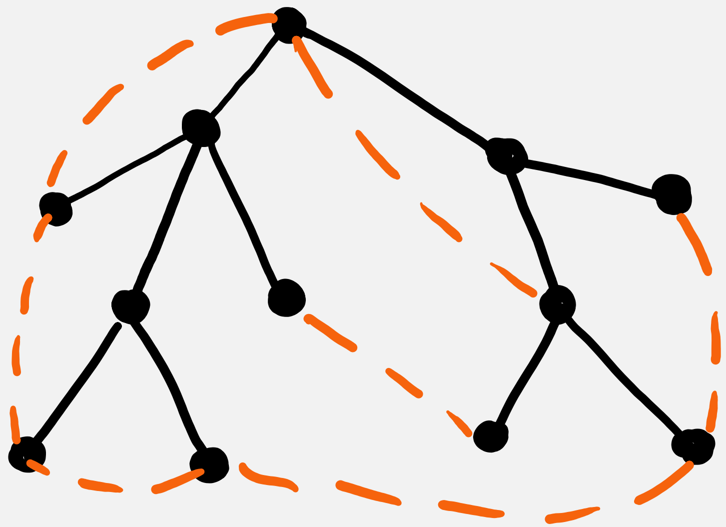
Problem Statement

Connectivity Augmentation Problem (k-CAP)

Input:

- $(k-1)$-edge-connected graph $G=(V,E)$, and
- set of weighted links L, where weights are in $\{0,1,\dots,W\}$

Output: min-weight $L' \subset L$ s.t. $G'=(V,E\cup L')$ is k-edge connected



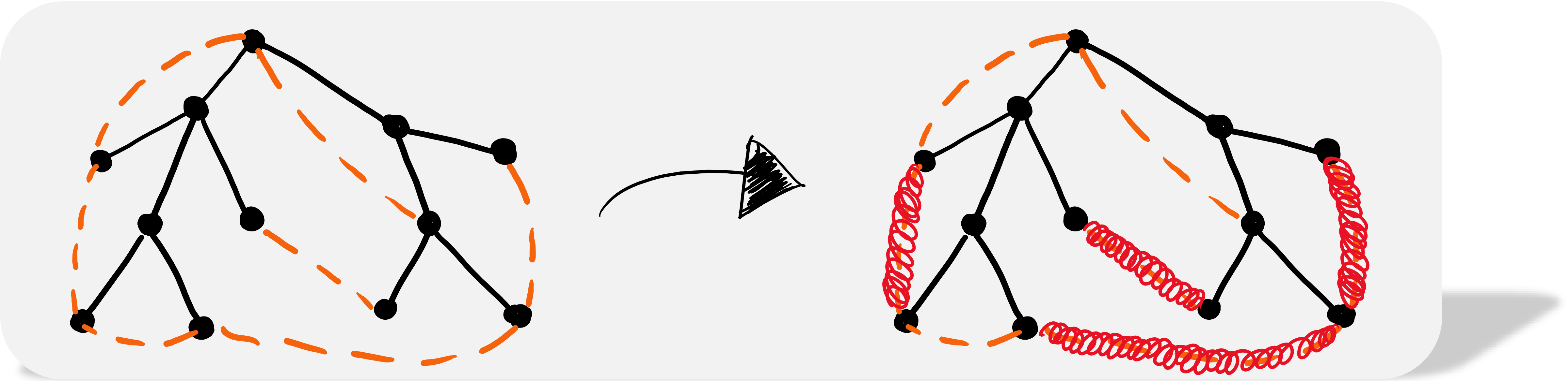
Problem Statement

Connectivity Augmentation Problem (k-CAP)

Input:

- $(k-1)$-edge-connected graph $G = (V, E)$, and
- set of weighted links L, where weights are in $\{0,1, \ldots, W\}$

Output: min-weight $L' \subset L$ s.t. $G' = (V, E \cup L')$ is k-edge connected



Problem Statement

Connectivity Augmentation Problem (k-CAP)

Input:

- $(k-1)$-edge-connected graph $G = (V, E)$, and
- set of weighted links L, where weights are in $\{0,1,\dots,W\}$

Output: min-weight $L' \subset L$ s.t. $G' = (V, E \cup L')$ is k-edge connected

➢ **Minimum Spanning Tree** and **Tree Augmentation Problem (TAP)** are special cases.

$k = 1$

$k = 2$

Problem Statement

Connectivity Augmentation Problem (k-CAP)

Input:

- $(k-1)$-edge-connected graph $G=(V,E)$, and
- set of weighted links L, where weights are in $\{0,1,\ldots,W\}$

Output: min-weight $L' \subset L$ s.t. $G' = (V, E \cup L')$ is k-edge connected

General Network Design Problem (aka SNDP)

k-ECSS:
$r(uv) = k$ for all $u, v \in V$

Input:

- graph $G=(V,E)$ with a weight function $w: E \to \{0,\ldots,W\}$, and
- connectivity requirement $r: V \times V \to \mathbb{Z}_{\geq 0}$

Output: min-weight $H \subset G$ s.t. $\forall s,t \in V$, H contains $r(st)$ edge-disjoint st-paths

Motivations

- Wireless/Telecommunication Networks
- Transportation Networks

Motivations

- Wireless/Telecommunication Networks
- Transportation Networks

- Its study led to fundamental theoretical advances in combinatorial optimization, algorithms and mathematical programming:
  - Primal-Dual
  - Iterative Rounding
- Recent Advancements on TAP and CAP [Byrka, Grandoni, Jabal Ameli'20] [Cecchetto, Traub, Zenklusen'21] [Traub, Zenklusen'22a,b] [Traub, Zenklusen'23] [Garg, Grandoni, Jabal Ameli'23]

What is Known (Offline)?

- **General Network Design Problem:**
  - ❑ 2-approximation [Jain'98]
- **Connectivity Augmentation Problem**
  - ❑ **Unweighted:** 1.393-approximation [Cecchetto, Traub, Zenklusen'21]
  - ❑ **Weighted:** $(1.5 + \epsilon)$-approximation [Traub, Zenklusen'23]
  - ❑ **Hardness:** APX-hard

Preliminaries: Streaming Model

Graph Streaming Model

- graph edges arrive in a stream, one by one (in an arbitrary order)
- using sublinear space, $O(n \operatorname{polylog}(n))$ space (known as **semi-streaming**)

Network Design in Streaming: increasing reliability of large-scale networks

- unlike graph problems such as MST, Matching, Cut, Sparsifiers, not much is known
- testing connectivity

Streaming Models: **Possible Computation Models**

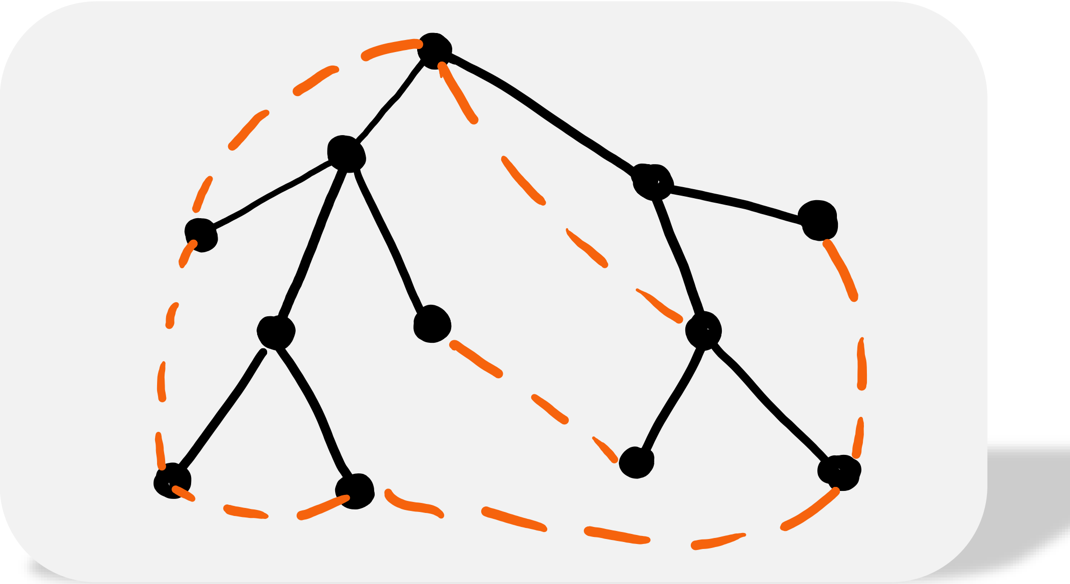
Connectivity Augmentation Problem

- Link Arrival Streams
  - ❑ G is given to the algorithm (its space is not counted in the space complexity)
  - ❑ links L arrive in a stream, one by one

Streaming Models: **Possible Computation Models**

Connectivity Augmentation Problem

- Link Arrival Streams
  - ❑ G is given to the algorithm (its space is not counted in the space complexity)
  - ❑ links L arrive in a stream, one by one



Streaming Models: Possible Computation Models

Connectivity Augmentation Problem

- Link Arrival Streams
  - ❑ G is given to the algorithm (its space is not counted in the space complexity)
  - ❑ links L arrive in a stream, one by one
  - ❑ compact representation exists:
    - o k-edge-connectivity certificate in $O(nk)$ space
    - o cactus representation of min-cuts in $O(n)$ space

Streaming Models: Possible Computation Models

Connectivity Augmentation Problem

- Link Arrival Streams
  - ❑ G is given to the algorithm (its space is not counted in the space complexity)
  - ❑ links L arrive in a stream, one by one
  - ❑ compact representation exists:
    - k-edge-connectivity certificate in $O(nk)$ space
    - cactus representation of min-cuts in $O(n)$ space
- (Fully) Edge Arrival Streams
  - ❑ both G and L arrive in a stream, in an arbitrary order

Streaming Models: **Possible Computation Models**

Connectivity Augmentation Problem

- Link Arrival Streams
  - ❑ G is given to the algorithm (its space is not counted in the space complexity)
  - ❑ links L arrive in a stream, one by one
  - ❑ compact representation exists:
    - o k-edge-connectivity certificate in $O(nk)$ space
    - o cactus representation of min-cuts in $O(n)$ sp

For General Network Design Problem, the only relevant model is **edge arrival**

- (Fully) Edge Arrival Streams
  - ❑ both G and L arrive in a stream, in an arbitrary order

Results

k-CAP, General Network Design, and Spanners

Our Results I: Weighted Connectivity Augmentation

	Approximation	Space	Model
k-CAP in one pass	$2+\epsilon$ $2-\epsilon$	$O(\frac{n}{\epsilon}\log n)$ $\Omega(n^2)$ bits	Link Arrival

- To get any approximation, $\Omega(n)$ space is needed.

Unlike the offline setting, the 2-approximation is a barrier in link arrival streams

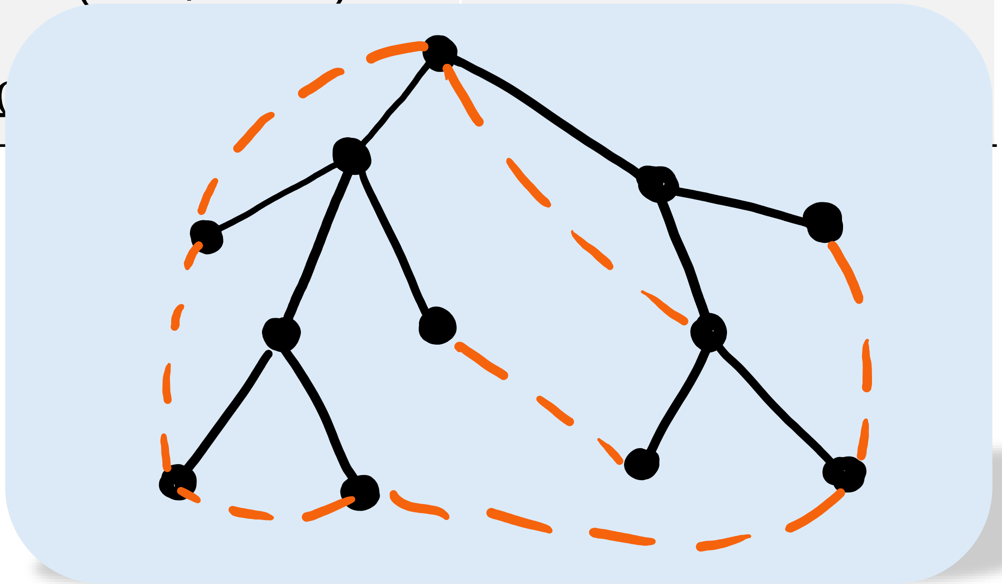
Our Results I: Weighted Connectivity Augmentation

	Approximation	Space	Model
k-CAP in one pass	$2+\epsilon$	$O(\frac{n}{\epsilon}\log n)$	Link Arrival
	$2-\epsilon$	$\Omega(n^2)$ bits	
	$O(t)$	$\tilde{O}(nk + n^{1+\frac{1}{t}})$	Edge Arrival
		$\Omega(nk + n^{1+\frac{1}{t}})$ bits	

- **An Interesting Special Case:** All links arrive before existing (zero cost) edges

Our Results I: Weighted Connectivity Augmentation

	Approximation	Space	Model
k-CAP in one pass	$2+\epsilon$ $2-\epsilon$	$O(\frac{n}{\epsilon}\log n)$ $\Omega(n^2)$ bits	Link Arrival
	$O(t)$	$\tilde{O}(nk + n^{1+\frac{1}{t}})$	



- **An Interesting Special Case:** All links arrive

Our Results I: Weighted Connectivity Augmentation

	Approximation	Space	Model
k-CAP in one pass	$2+\epsilon$	$O(\frac{n}{\epsilon}\log n)$	Link Arrival
	$2-\epsilon$	$\Omega(n^2)$ bits	
	$O(t)$	$\tilde{O}(nk + n^{1+\frac{1}{t}})$	Edge Arrival
		$\Omega(nk + n^{1+\frac{1}{t}})$ bits	

- **An Interesting Special Case:** All links arrive before existing (zero cost) edges
- nk denotes the amount of space required to construct the k-connectivity certificate

Spanner is **the sparsifier** for connectivity augmentation problem.

Our Results II: (Weighted) Spanners

- Tight bound in unweighted case: $O(t)$-spanners with $O(n^{1+\frac{1}{t}})$ edges
- **Weighted Case:** Existing methods give $O(t)$-spanners with $O(n^{1+\frac{1}{t}} \cdot \log W)$ edges

	Distortion	Space	Model
Weighted Spanners in one pass	$O(t)$	$\tilde{O}(n^{1+\frac{1}{t}})$ $\Omega(n^{1+\frac{1}{t}})$ bits	Edge Arrival

Results III: Applications to General Network Design

	Passes	Approximation	Space	Model
SNDP	1	$O(t \log k)$ $O(t)$	$\tilde{O}(kn^{1+\frac{1}{t}})$ $\Omega(n^{1+\frac{1}{t}})$ bits	Edge Arrival
k-ECSS	k	$O(\log k)$	$O(nk \log n)$	Edge Arrival

Algorithms Overview

Weighted Spanners / k-CAP in Streams

Weighted Spanner Problem

Input:

- edge-weighted graph $G = (V, E)$ where weights belong to $\{1, \dots, W\}$,
- distortion parameter t

Output: subgraph $H \subset G$ s.t. for every $u, v \in V$, $d_H(u, v) \leq t \cdot d_G(u, v)$

Weighted Spanner Problem

Input:

- edge-weighted graph $G = (V, E)$ where weights belong to $[1, W]$,
- distortion parameter t

Output: subgraph $H \subset G$ s.t. for every $u, v \in V$, $d_H(u, v) \leq t \cdot d_G(u, v)$

Algorithm for Unweighted Graphs:

1. Initialize $H = (V, E')$ with $E' = \emptyset$
2. For every edge $e \in E$, add e to H if it does not form a cycle of length $\leq O(t)$ with existing edges in H

Simple streaming-friendly algorithm with space complexity $n^{1+1/t}$

Weighted Spanner Problem

Input:

- edge-weighted graph $G = (V, E)$ where weights belong to $[1, W]$,
- distortion parameter t

Output: subgraph $H \subset G$ s.t. for every $u, v \in V$, $d_H(u, v) \leq t \cdot d_G(u, v)$

Simple Extension of Algorithm for Weighted Graphs:

- Partition edges into $\log W$ classes s.t. edges in each class differ by $O(1)$-factor
- Maintain $O(t)$-spanner in each weight class

Simple streaming-friendly algorithm
with space complexity $O(n^{1+\frac{1}{t}} \cdot \log W)$

Shaving the $\log W$ Factor: Even-Odd Bucketing

Standard Partitioning (into $\log W$ partitions)

Even-Odd Bucketing

$$\cdots \quad \underbrace{[n^{i-2}, n^{i-1})}_{B_{i-2}} \quad \underbrace{[n^{i-1}, n^{i})}_{B_{i-1}} \quad \underbrace{[n^{i}, n^{i+1})}_{B_{i}} \quad \underbrace{[n^{i+1}, n^{i+2})}_{B_{i+1}} \quad \underbrace{[n^{i+2}, n^{i+3})}_{B_{i+2}} \quad \cdots$$

Key Property: It is cheaper to pick n edges from B_{i-2} than to pick a single edge from B_i

- ❑ Separately maintain spanners on **even buckets** and **odd buckets**

Shaving the $\log W$ Factor: Even-Odd Bucketing

Standard Partitioning (into $\log W$ partitions)

Even-Odd Bucketing

$\cdots$ $[n^{i-2}, n^{i-1})$ $[n^{i-1}, n^{i})$ $[n^{i}, n^{i+1})$ $[n^{i+1}, n^{i+2})$ $[n^{i+2}, n^{i+3})$ $\cdots$

B_{i-2} B_{i-1} B_i B_{i+1} B_{i+2}

Key Property: It is cheaper to pick n edges from B_{i-2} than to pick a single edge from B_i

- ❑ Separately maintain spanners on **even buckets** and **odd buckets**

Shaving the $\log W$ Factor: Even-Odd Bucketing

Standard Partitioning (into $\log W$ partitions)

Even-Odd Bucketing

… $[n^{i-2}, n^{i-1})$ B_{i-2} | $[n^{i-1}, n^{i})$ B_{i-1} | $[n^{i}, n^{i+1})$ B_{i} | $[n^{i+1}, n^{i+2})$ B_{i+1} | $[n^{i+2}, n^{i+3})$ B_{i+2} …

Key Property: It is cheaper to pick n edges from B_{i-2} than to pick a single edge from B_i

❑ Separately maintain spanners on **even buckets** and **odd buckets**

Outline of Weighted Spanner Algorithm

Step 1. Summarize edges in **even** and **odd** buckets separately **(maintaining a $O(t)$-spanner)**

Step 2. Combine t-spanners of **even** and **odd** buckets

Algorithm for Spanners on Even Buckets

… $[n^{i-2}, n^{i-1})$ $[n^{i-1}, n^{i})$ $[n^{i}, n^{i+1})$ $[n^{i+1}, n^{i+2})$ $[n^{i+2}, n^{i+3})$ …

B_{i-2} B_{i-1} B_i B_{i+1} B_{i+2}

- For any edge $uv \in B_i$:
  - If there exists a uv-path in (even buckets of) $B_{\leq i-2}$, don't keep uv in spanner

Key Property: For every ***uv*-edge** in B_i, a ***uv*-path** (of any length) in $B_{\leq i-2}$ (if exists) has a lower weight

Algorithm for Spanners on Even Buckets

… $[n^{i-2}, n^{i-1})$ $[n^{i-1}, n^{i})$ $[n^{i}, n^{i+1})$ $[n^{i+1}, n^{i+2})$ $[n^{i+2}, n^{i+3})$ …

$\boldsymbol{B_{i-2}}$ $\boldsymbol{B_{i-1}}$ $\boldsymbol{B_{i}}$ $\boldsymbol{B_{i+1}}$ $\boldsymbol{B_{i+2}}$

- For any edge $uv \in \boldsymbol{B_i}$:
  - If there exists a uv-path in (even buckets of) $\boldsymbol{B_{\leq i-2}}$, don't keep uv in spanner
- Need to deal with edges of $\boldsymbol{B_i}$ between different CCs of $G[\boldsymbol{B_{\leq i-2}}]$

Algorithm for Spanners on Even Buckets

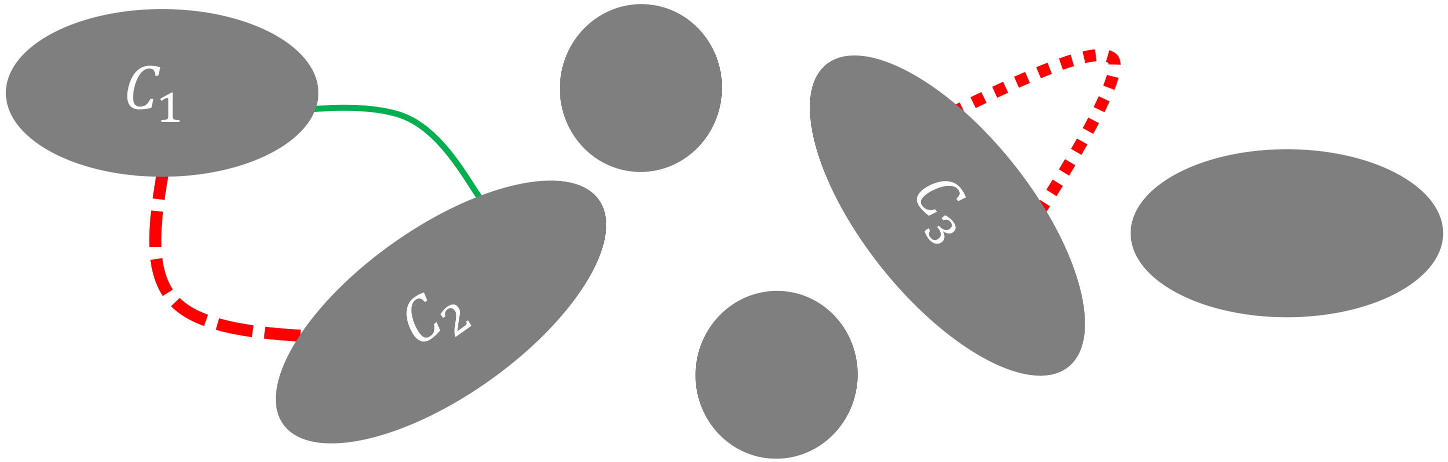
… $[n^{i-2}, n^{i-1})$ $[n^{i-1}, n^{i})$ $[n^{i}, n^{i+1})$ $[n^{i+1}, n^{i+2})$ $[n^{i+2}, n^{i+3})$ …

$\boldsymbol{B}_{i-2}$ $\boldsymbol{B}_{i-1}$ $\boldsymbol{B}_{i}$ $\boldsymbol{B}_{i+1}$ $\boldsymbol{B}_{i+2}$

- For any edge $uv \in \boldsymbol{B}_i$:
  - If there exists a uv-path in (even buckets of) $\boldsymbol{B}_{\leq i-2}$, don't keep uv in spanner
- Need to deal with edges of $\boldsymbol{B}_i$ between different CCs of $G[\boldsymbol{B}_{\leq i-2}]$

I. Remove edges in bucket i with both endpoints in a same CC

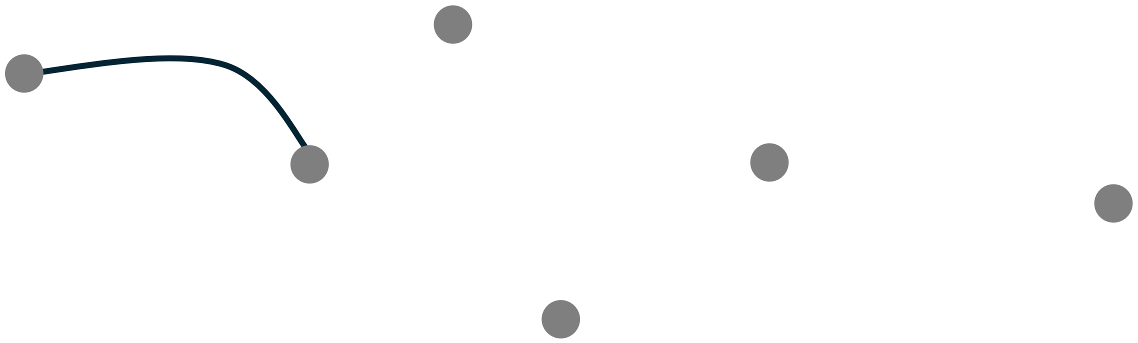
II. Between any pair of CC, only keep the lightest edge from bucket i



Algorithm for Spanners on Even Buckets

…	$[n^{i-2}, n^{i-1})$	$[n^{i-1}, n^{i})$	$[n^{i}, n^{i+1})$	$[n^{i+1}, n^{i+2})$	$[n^{i+2}, n^{i+3})$	…
	$\boldsymbol{B}_{i-2}$	$\boldsymbol{B}_{i-1}$	$\boldsymbol{B}_{i}$	$\boldsymbol{B}_{i+1}$	$\boldsymbol{B}_{i+2}$	

- For any edge $uv \in \boldsymbol{B}_i$:
  - If there exists a uv-path in (even buckets of) $\boldsymbol{B}_{\leq i-2}$, don't keep uv in spanner
- Need to deal with edges of $\boldsymbol{B}_i$ between different CCs of $G[\boldsymbol{B}_{\leq i-2}]$
- Contract CCs of $G[\boldsymbol{B}_{\leq i-2}]$ into super nodes



Analysis of the Algorithm on Even Buckets

… $[n^{i-2}, n^{i-1})$ $[n^{i-1}, n^{i})$ $[n^{i}, n^{i+1})$ $[n^{i+1}, n^{i+2})$ $[n^{i+2}, n^{i+3})$ …

$\boldsymbol{B_{i-2}}$ $\boldsymbol{B_{i-1}}$ $\boldsymbol{B_{i}}$ $\boldsymbol{B_{i+1}}$ $\boldsymbol{B_{i+2}}$

- If there are x super nodes in $\boldsymbol{B_i}$, need to store at most $\tilde{O}(x^{1+1/t})$ edges from $\boldsymbol{B_i}$
  - Using the simple weighted spanner algorithm within each $\boldsymbol{B_i}$ (increase the space complexity by $\log n$)
  - If the number of non-zero-degree super nodes w.r.t. $\boldsymbol{B_i}$ is y, the spanner size becomes $\tilde{O}(y^{1+1/t})$
- Using the nested structure of CCs w.r.t. buckets ($\boldsymbol{B_{\leq i}}, \boldsymbol{B_{\leq i-2}}, \ldots$), we can bound the total number of edges by $\tilde{O}(n^{1+1/t})$

Implementation in Streaming

$\cdots \quad [n^{i-2}, n^{i-1}) \quad [n^{i-1}, n^{i}) \quad [n^{i}, n^{i+1}) \quad [n^{i+1}, n^{i+2}) \quad [n^{i+2}, n^{i+3}) \quad \cdots$

$\quad\quad \boldsymbol{B_{i-2}} \quad\quad \boldsymbol{B_{i-1}} \quad\quad \boldsymbol{B_{i}} \quad\quad \boldsymbol{B_{i+1}} \quad\quad \boldsymbol{B_{i+2}}$

- Edges may come in any order and CCs may change for several buckets as we add an edge

- Maintain super nodes in the stream (given the set of so far stored edges H, we can compute them when needed)
- Sparsify the spanners of all heavier buckets once an edge is added to H

Algorithms Overview

k-CAP in Streams

Weighted k-CAP in Streams

- Edge Arrival Stream

  Immediately follows from our streaming algorithm for **weighted spanner**

- Link Arrival Stream

  It uses **even-odd bucketing**, but needs to deal with more involved structures

Overview of k-CAP in Link Arrival

2-Approximation Algorithm in Link Arrival

Connectivity Augmentation → Cactus Augmentation → Cycle Augmentation

[Galvez, Grandoni, Jabal Ameli, Sornat'19]
[Traub, Zenklusen'23]

Overview of k-CAP in Link Arrival

2-Approximation Algorithm in Link Arrival

Connectivity Augmentation → Cactus Augmentation → Cycle Augmentation

- **Warm-up:** using extra $\log W$ factor in space compared to the unweighted case:

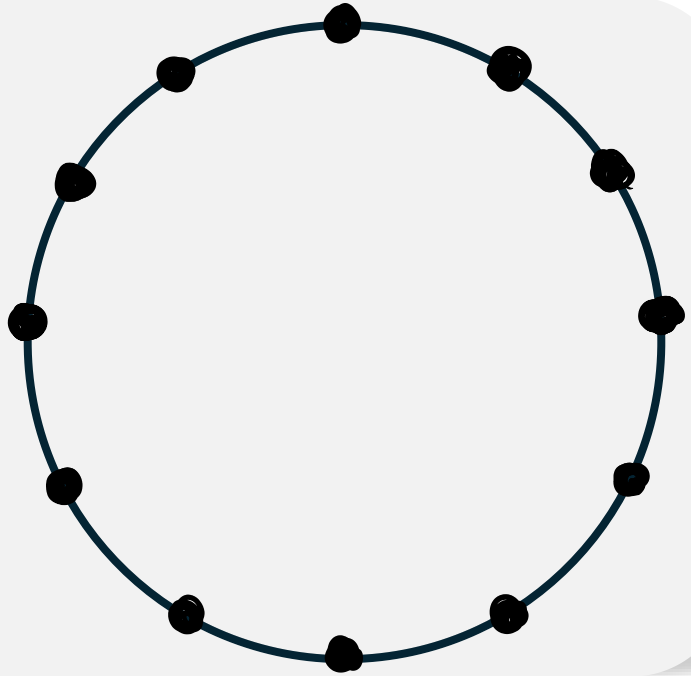
Overview of k-CAP in Link Arrival

2-Approximation Algorithm in Link Arrival

Connectivity Augmentation → Cactus Augmentation → Cycle Augmentation

- **Warm-up:** using extra $\log W$ factor in space compared to the unweighted case:

Cycle Augmentation



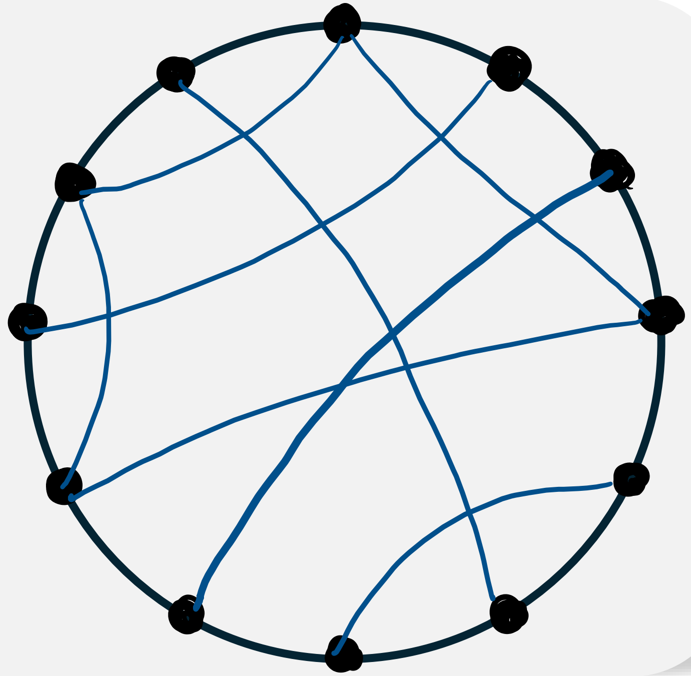
Overview of k-CAP in Link Arrival

2-Approximation Algorithm in Link Arrival

Connectivity Augmentation → Cactus Augmentation → Cycle Augmentation

- **Warm-up:** using extra $\log W$ factor in space compared to the unweighted case:

Cycle Augmentation



Overview of k-CAP in Link Arrival

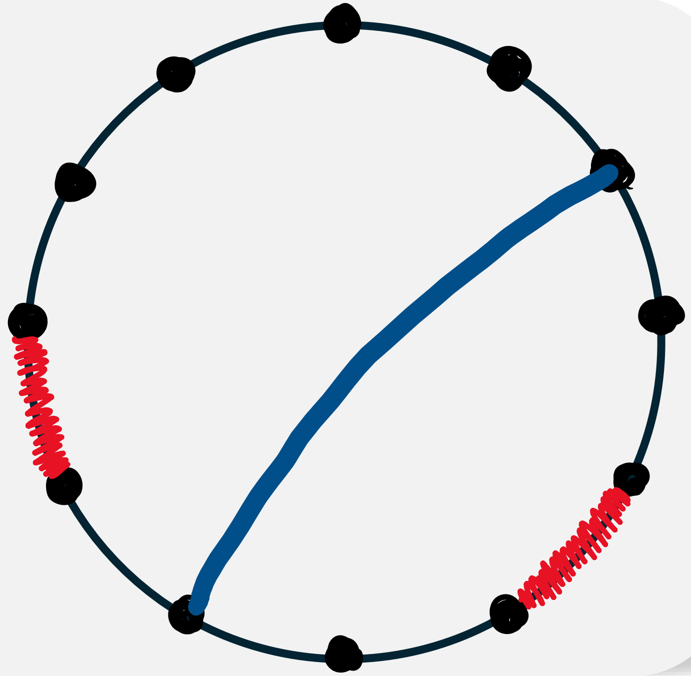
2-Approximation Algorithm in Link Arrival

Connectivity Augmentation → Cactus Augmentation → Cycle Augmentation

- **Warm-up:** using extra $\log W$ factor in space compared to the unweighted case:

Cycle Augmentation

- 2-cuts corresponds to removal of two edges on the cycle
  - A 2-cut is covered if there exists a crossing link



Overview of k-CAP in Link Arrival

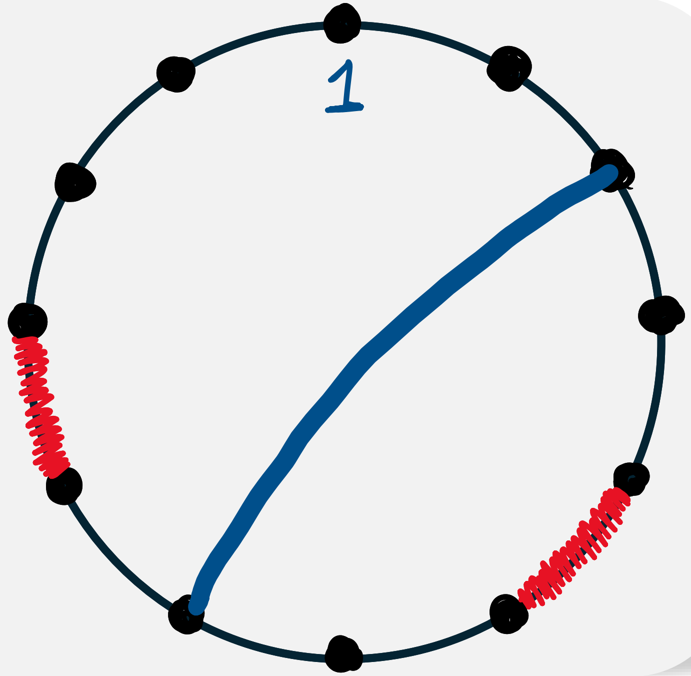
2-Approximation Algorithm in Link Arrival

Connectivity Augmentation → **Cactus Augmentation** → **Cycle Augmentation**

- **Warm-up:** using extra $\log W$ factor in space compared to the unweighted case:

Cycle Augmentation

- 2-cuts corresponds to removal of two edges on the cycle
  - A 2-cut is covered if there exists a crossing link
- Pick an arbitrary root node "**1**"



Overview of k-CAP in Link Arrival

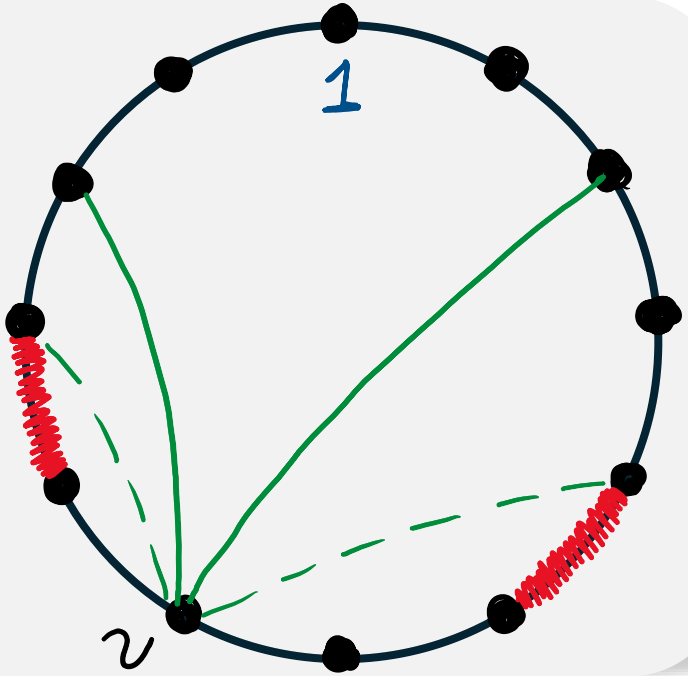
2-Approximation Algorithm in Link Arrival

Connectivity Augmentation → **Cactus Augmentation** → **Cycle Augmentation**

- **Warm-up:** using extra $\log W$ factor in space compared to the unweighted case:

Cycle Augmentation

- 2-cuts corresponds to removal of two edges on the cycle
  - A 2-cut is covered if there exists a crossing link
- Pick an arbitrary root node "**1**"
- Keep incident edges of v whose endpoints are closets to "**1**"
  - Total of $O(n)$ edges are kept.

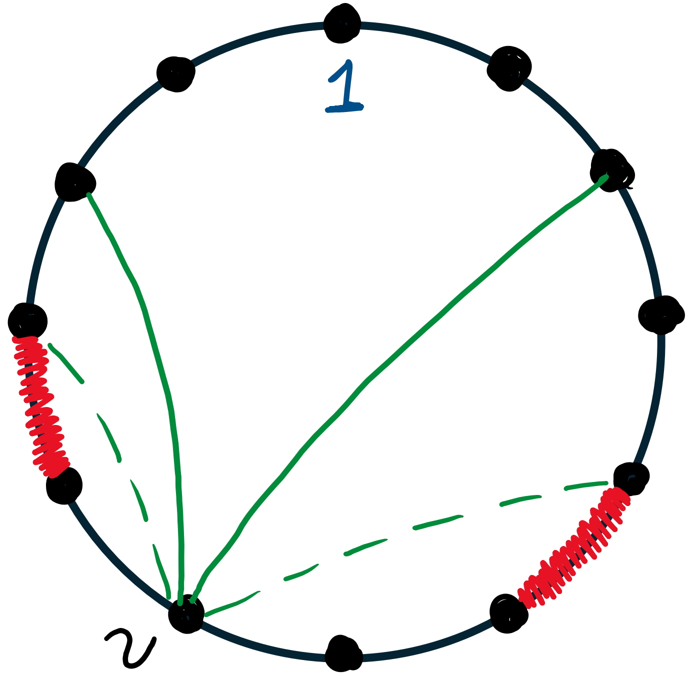


Overview of k-CAP in Link Arrival

2-Approximation Algorithm in Link Arrival

Connectivity Augmentation → **Cactus Augmentation** → **Cycle Augmentation**

- **Warm-up:** using extra $\log W$ factor in space compared to the unweighted case:
  - ❑ keep at most 2 edges per vertex for each weight class
- **Weighted:** similar **even-odd bucketing** idea works
  - ❑ shrink "3-edge-connected" components
  - ❑ not Cycle Augmentation anymore
  - ❑ still has nice structures to exploit



Algorithms Overview

General Network Design in Streams

General Network Design

k-Edge Connected Spanning Subgraph (k-ECSS)

- Augmentation Framework [Williamson et al.'93]
  - ❑ Increase the connectivity one by one in each pass till get to k
  - ❑ Using our algorithm for CAP in link arrival, we get an algorithm that runs
    - in k passes,
    - uses $O(nk + n \cdot \log n)$ space, and
    - **Approx. factor**: trivial bound is $O(k)$ but a more careful analysis gives $O(\log k)$

Approximation Bound

- The integrality gap of the standard LP relaxation of the augmentation problem is 2.
- In ℓ-th phase (i.e., ℓ-CAP), the optimal fractional solution has cost $\leq \frac{\mathrm{OPT}_{k-\mathrm{ECSS}}}{\ell}$

General Network Design (contd.)

SNDP: A collection of k spanner-like structure is a coreset for the problem

- Analysis relies on
  - **Reverse** Augmentation Framework [Goemans et al.'94], and
  - Our algorithm for CAP in edge arrival streams
- Guarantee
  - In one pass, and
  - using $O(kn^{1+1/t})$,
  - achieves $O(t \cdot \log k)$-approximation

Summary:

"Network Design in streaming and its connection to weighted spanners"

Questions

- Close the gaps (for SNDP and k-ECSS)!
- Close the gaps for unweighted k-CAP in link arrival streams:
  - UB: $(2+\epsilon)$-approx. in $\tilde{O}(n)$ space
  - LB: < 1.409-approx. not possible in $\tilde{O}(n)$ space
- Dynamic streaming?

Problem	Pass	Approx.	Space	Stream
k-CAP	1	$2+\epsilon$ $2-\epsilon$	$O(\frac{n}{\epsilon}\log n)$ $\Omega(n^2)$ bits	link arrival
		$O(t)$	$\tilde{O}(kn + n^{1+\frac{1}{t}})$ $\Omega(kn + n^{1+\frac{1}{t}})$ bits	fully streaming
STAP	1	$O(t)$	$\tilde{O}(n^{1+\frac{1}{t}})$ $\Omega(n^{1+\frac{1}{t}})$ bits	fully streaming link arrival
Spanner	1	$O(t)$	$\tilde{O}(n^{1+\frac{1}{t}})$ $\Omega(n^{1+\frac{1}{t}})$ bits	edge arrival
SNDP	1	$O(t\log k)$ $O(t)$	$\tilde{O}(kn^{1+\frac{1}{t}})$ $\Omega(n^{1+\frac{1}{t}})$ bits	edge arrival
k-ECSS	k	$O(\log k)$	$O(kn\log n)$	edge arrival

Thank you!