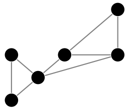
# Tight Bounds for Vertex Connectivity in Dynamic Streams

Sepehr Assadi & Vihan Shah

Rutgers University

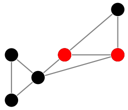
# Vertex Connectivity

- Undirected Graph $G = (V, E)$
- Vertex Connectivity: Minimum number of vertices that need to be deleted to disconnect $G$



# Vertex Connectivity

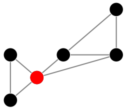
- Undirected Graph $G = (V, E)$
- Vertex Connectivity: Minimum number of vertices that need to be deleted to disconnect $G$



2

# Vertex Connectivity

- Undirected Graph $G = (V, E)$
- Vertex Connectivity: Minimum number of vertices that need to be deleted to disconnect $G$



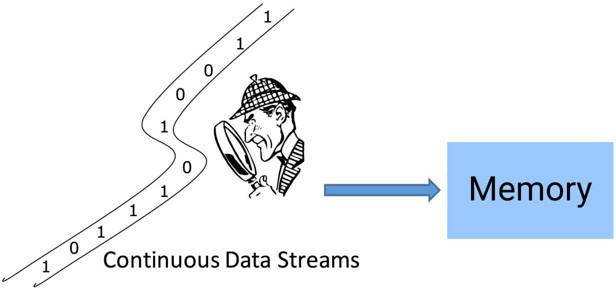
1

# Classical Setting

- Can find vertex connectivity in $\mathrm{polylog}\, m$ max flow time [LNP+21]
- Recent breakthrough for max flow: $m^{1+o(1)}$ time [CKL+22]
- Thus, finding vertex connectivity also takes $m^{1+o(1)}$ time

# Graph Streaming

- $G = (V, E)$
- Edges of $G$ appear in a stream
- Trivial Solution: Store all edges ($\Omega(n^2)$ space)
- Goal: Minimize Memory ($o(n^2)$ space)

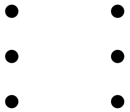


# Streaming Models

**Insertion-Only**

# Streaming Models

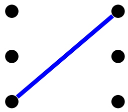
**Insertion-Only**



# Streaming Models

**Insertion-Only**

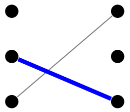
$\boxed{e_1}$



# Streaming Models

## Insertion-Only

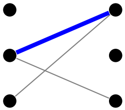
| | |
|---|---|
| $e_1$ | $e_2$ |



# Streaming Models

## Insertion-Only

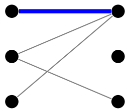
| | | |
|---|---|---|
| $e_1$ | $e_2$ | $e_3$ |



# Streaming Models

**Insertion-Only**

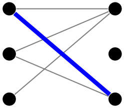
| $e_1$ | $e_2$ | $e_3$ | $e_4$ |
|---|---|---|---|



# Streaming Models

## Insertion-Only

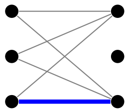
| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ |
|---|---|---|---|---|



# Streaming Models

**Insertion-Only**

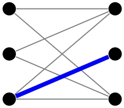
| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ |
|---|---|---|---|---|---|



# Streaming Models

**Insertion-Only**

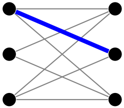
| | | | | | | |
|---|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ |



# Streaming Models

## Insertion-Only

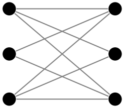
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |



# Streaming Models

**Insertion-Only** (finite stream)

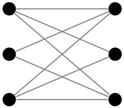
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |



# Streaming Models

**Insertion-Only** (finite stream)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |

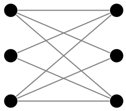


**Dynamic**

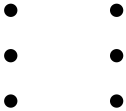
# Streaming Models

**Insertion-Only** (finite stream)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |



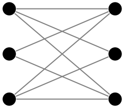
**Dynamic**



# Streaming Models

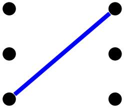
**Insertion-Only** (finite stream)

| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |
|---|---|---|---|---|---|---|---|



**Dynamic**

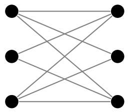
| $e_1$ |
|---|



# Streaming Models

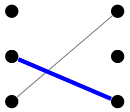
**Insertion-Only** (finite stream)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |



**Dynamic**

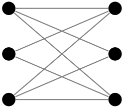
| | |
|---|---|
| $e_1$ | $e_2$ |



# Streaming Models

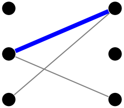
**Insertion-Only** (finite stream)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |



**Dynamic**

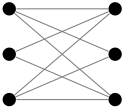
| | | |
|---|---|---|
| $e_1$ | $e_2$ | $e_3$ |



# Streaming Models

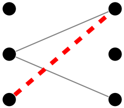
**Insertion-Only** (finite stream)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |



**Dynamic**

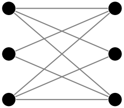
| | | | |
|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $\overline{e_1}$ |



# Streaming Models

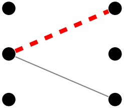
**Insertion-Only** (finite stream)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |



**Dynamic**

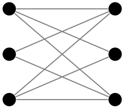
| | | | | |
|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $\overline{e_1}$ | $\overline{e_3}$ |



# Streaming Models

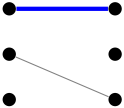
**Insertion-Only** (finite stream)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |



**Dynamic**

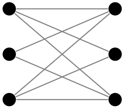
| | | | | | |
|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $\overline{e_1}$ | $\overline{e_3}$ | $e_4$ |



# Streaming Models

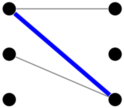
**Insertion-Only** (finite stream)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |



**Dynamic**

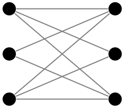
| | | | | | | |
|---|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $\overline{e_1}$ | $\overline{e_3}$ | $e_4$ | $e_5$ |



# Streaming Models

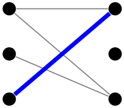
**Insertion-Only** (finite stream)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |



**Dynamic**

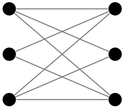
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $\overline{e_1}$ | $\overline{e_3}$ | $e_4$ | $e_5$ | $e_1$ |



# Streaming Models

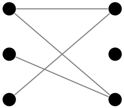
**Insertion-Only** (finite stream)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |



**Dynamic** (finite stream)

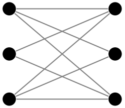
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $\overline{e_1}$ | $\overline{e_3}$ | $e_4$ | $e_5$ | $e_1$ |



# Streaming Models

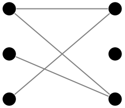
**Insertion-Only** (finite stream)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |



**Dynamic** (finite stream)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $\overline{e_1}$ | $\overline{e_3}$ | $e_4$ | $e_5$ | $e_1$ |



We want to solve the problem after a single pass of the stream

# Our Problem

- Finding exact vertex connectivity needs $\Omega(n^2)$ space in the worst case [SW15]

# Our Problem

- Finding exact vertex connectivity needs $\Omega(n^2)$ space in the worst case [SW15]
- We want to solve the $k$-vertex connectivity problem in streaming (is the vertex connectivity of the input graph $G < k$ or $\geq k$)

# Our Problem

- Finding exact vertex connectivity needs $\Omega(n^2)$ space in the worst case [SW15]
- We want to solve the $k$-vertex connectivity problem in streaming (is the vertex connectivity of the input graph $G < k$ or $\geq k$)
- We also want to output a certificate of connectivity (If $G$ is $k$-vertex connected, output a subgraph $H$ (certificate) that is also $k$-vertex connected)

# Previous Work

**Insertion-Only**

1. Upper bound: $\widetilde{O}(kn)$ [FKM+05]
2. Lower bound: $\Omega(kn)$ [SW15]

# Previous Work

**Insertion-Only**

1. Upper bound: $\widetilde{O}(kn)$ [FKM+05]
2. Lower bound: $\Omega(kn)$ [SW15]

**Dynamic**

1. Upper bound: $\widetilde{O}(k^2 n)$ [GMT15]
2. Lower bound: $\Omega(kn)$ [SW15]

# Previous Work

**Insertion-Only**

1. Upper bound: $\widetilde{O}(kn)$ [FKM+05]
2. Lower bound: $\Omega(kn)$ [SW15]

**Dynamic**

1. Upper bound: $\widetilde{O}(k^2 n)$ [GMT15]
2. Lower bound: $\Omega(kn)$ [SW15]

The lower bounds hold even when a certificate is not needed

# Previous Work

**Insertion-Only**

1. Upper bound: $\widetilde{O}(kn)$ [FKM+05]
2. Lower bound: $\Omega(kn)$ [SW15]

**Dynamic**

1. Upper bound: $\widetilde{O}(k^2 n)$ [GMT15]
2. Lower bound: $\Omega(kn)$ [SW15]

There is a gap of factor $k$ between the best known upper and lower bound in dynamic streams

# Insertion-Only vs Dynamic Streams

# Insertion-Only vs Dynamic Streams

- Most graph problems studied in insertion-only streams, have similar guarantees in dynamic streams
- Examples: Connectivity [AGM12a], Cut Sparsifiers [AGM12b], Subgraph Counting [AGM12b], $(\Delta + 1)$-Vertex Coloring [ACK19]

# Insertion-Only vs Dynamic Streams

- Most graph problems studied in insertion-only streams, have similar guarantees in dynamic streams
- Examples: Connectivity [AGM12a], Cut Sparsifiers [AGM12b], Subgraph Counting [AGM12b], $(\Delta + 1)$-Vertex Coloring [ACK19]
- However for Matching and Vertex Cover a 2-approximation in insertion-only streams takes space $\widetilde{O}(n)$ but an $O(1)$-approximation in dynamic streams needs $\Omega(n^2)$ space

# Insertion-Only vs Dynamic Streams

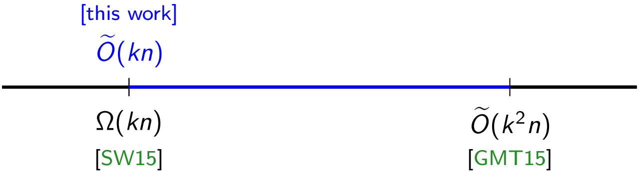
- Most graph problems studied in insertion-only streams, have similar guarantees in dynamic streams
- Examples: Connectivity [AGM12a], Cut Sparsifiers [AGM12b], Subgraph Counting [AGM12b], $(\Delta + 1)$-Vertex Coloring [ACK19]
- However for Matching and Vertex Cover a $2$-approximation in insertion-only streams takes space $\widetilde{O}(n)$ but an $O(1)$-approximation in dynamic streams needs $\Omega(n^2)$ space
- It was unresolved which category vertex connectivity belonged to

# Our Results

We bridge the gap between the upper and lower bound in dynamic streams

**Theorem**

*There exists a randomized dynamic graph streaming algorithm for $k$-vertex connectivity that succeeds with high probability and uses $\widetilde{O}(kn)$ space.*

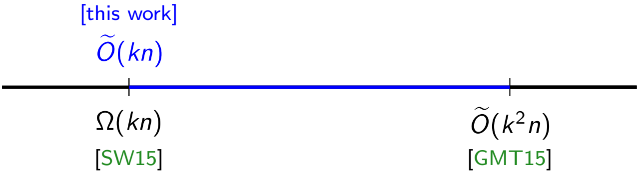


# Our Results

We bridge the gap between the upper and lower bound in dynamic streams

**Theorem**

*There exists a randomized dynamic graph streaming algorithm for $k$-vertex connectivity that succeeds with high probability and uses $\widetilde{O}(kn)$ space.*




Note: We also output a certificate of vertex connectivity

# Our Results

We also extend the lower bound of [SW15] to multiple pass streams:

**Theorem**

*Any randomized $p$-pass insertion-only streaming algorithm that solves the $k$-vertex connectivity problem with probability at least $2/3$ needs $\Omega(kn/p)$ bits of space.*

# Our Results

We also extend the lower bound of [SW15] to multiple pass streams:

**Theorem**

*Any randomized $p$-pass insertion-only streaming algorithm that solves the $k$-vertex connectivity problem with probability at least $2/3$ needs $\Omega(kn/p)$ bits of space.*

Note: This lower bound is for multi-graphs (also the case for [SW15])

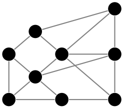
The upper bound also works for multi-graphs

# Algorithm of [GMT15]

For $i = 1$ to $r = O(k^2 \log n)$:

1. Sample every vertex in $V_i$ independently with probability $1/k$
2. Store a spanning forest $H_i$ on $G[V_i]$

Output $H = \cup_i H_i$ as the certificate



$G$

# Algorithm of [GMT15]

For $i = 1$ to $r = O(k^2 \log n)$:

1. Sample every vertex in $V_i$ independently with probability $1/k$
2. Store a spanning forest $H_i$ on $G[V_i]$

Output $H = \cup_i H_i$ as the certificate

# Algorithm of [GMT15]

For $i = 1$ to $r = O(k^2 \log n)$:

1. Sample every vertex in $V_i$ independently with probability $1/k$
2. Store a spanning forest $H_i$ on $G[V_i]$

Output $H = \cup_i H_i$ as the certificate

# Algorithm of [GMT15]

For $i = 1$ to $r = O(k^2 \log n)$:

1. Sample every vertex in $V_i$ independently with probability $1/k$
2. Store a spanning forest $H_i$ on $G[V_i]$

Output $H = \cup_i H_i$ as the certificate

$G$ $\quad$ $H_3$

# Algorithm of [GMT15]

For $i = 1$ to $r = O(k^2 \log n)$:

1. Sample every vertex in $V_i$ independently with probability $1/k$
2. Store a spanning forest $H_i$ on $G[V_i]$

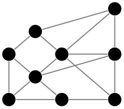
Output $H = \cup_i H_i$ as the certificate

# Algorithm of [GMT15]

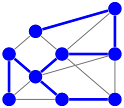
For $i = 1$ to $r = O(k^2 \log n)$:

1. Sample every vertex in $V_i$ independently with probability $1/k$
2. Store a spanning forest $H_i$ on $G[V_i]$

Output $H = \cup_i H_i$ as the certificate



$G$



$H$

# Guarantee of [GMT15]

[GMT15] proved the following:

- If $G$ is not $k$-connected then $H$ will not be $k$-connected
- If $G$ is $2k$-connected $H$ will be $k$-connected

# Guarantee of [GMT15]

[GMT15] proved the following:

- If $G$ is not $k$-connected then $H$ will not be $k$-connected
- If $G$ is $2k$-connected $H$ will be $k$-connected

So, this was an approximation algorithm for $k$-vertex connectivity.

# Guarantee of [GMT15]

[GMT15] proved the following:

- If $G$ is not $k$-connected then $H$ will not be $k$-connected
- If $G$ is $2k$-connected $H$ will be $k$-connected

So, this was an approximation algorithm for $k$-vertex connectivity.

We give a better analysis of the same algorithm and show that it works for exact $k$-vertex connectivity.

# Space

- Each sample has $n/k$ vertices in expectation

# Space

- Each sample has $n/k$ vertices in expectation
- The spanning forest algorithm takes $\widetilde{O}(n/k)$ space [AGM12a]

# Space

- Each sample has $n/k$ vertices in expectation
- The spanning forest algorithm takes $\widetilde{O}(n/k)$ space [AGM12a]
- Repeating $O(k^2 \log n)$ times gives a space bound of $\widetilde{O}(kn)$

# Space

- Each sample has $n/k$ vertices in expectation
- The spanning forest algorithm takes $\widetilde{O}(n/k)$ space [AGM12a]
- Repeating $O(k^2 \log n)$ times gives a space bound of $\widetilde{O}(kn)$
- Concentration bounds to get $\widetilde{O}(kn)$ space whp

# Key Properties

We have the following two key properties:

**Lemma (Property 1)**

*Every edge whose endpoints are less than $2k$ connected in $G$ exists in $H$ whp.*

**Lemma (Property 2)**

*Every pair of vertices that is at least $2k$ connected in $G$ is at least $k$ connected in $H$ whp. [GMT15]*

# Correctness

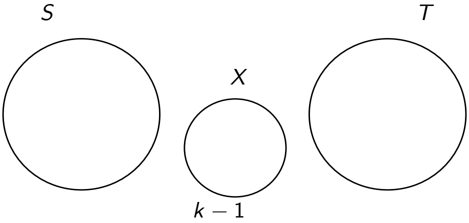
- If $G$ is not $k$-vertex-connected, $H$ is not $k$-vertex-connected

# Correctness

- If $G$ is not $k$-vertex-connected, $H$ is not $k$-vertex-connected
- Assume $G$ is $k$-vertex-connected but $H$ is not $k$-vertex-connected

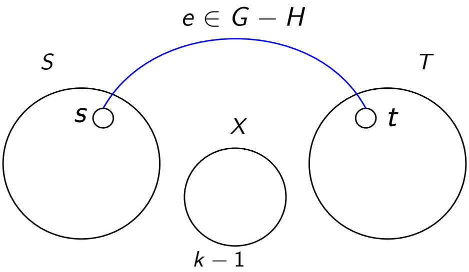
# Correctness

- If $G$ is not $k$-vertex-connected, $H$ is not $k$-vertex-connected
- Assume $G$ is $k$-vertex-connected but $H$ is not $k$-vertex-connected
- Deleting $X$ (of size at most $k-1$) disconnects $H$



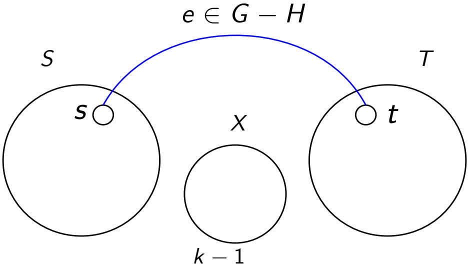
# Correctness

- If $G$ is not $k$-vertex-connected, $H$ is not $k$-vertex-connected
- Assume $G$ is $k$-vertex-connected but $H$ is not $k$-vertex-connected
- Deleting $X$ (of size at most $k-1$) disconnects $H$
- There is an edge between $S$ and $T$ in $G$ but not in $H$



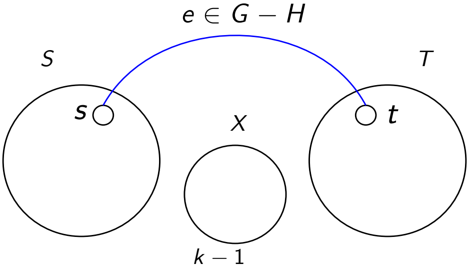
# Correctness

- Case 1: $s$ and $t$ have $< 2k$ vertex-disjoint paths between them in $G$



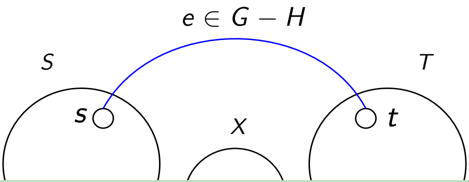
# Correctness

- Case 1: $s$ and $t$ have $< 2k$ vertex-disjoint paths between them in $G$
- The edge $(s, t)$ must be in $H$ (Property 1)



# Correctness

- Case 1: $s$ and $t$ have $< 2k$ vertex-disjoint paths between them in $G$
- The edge $(s, t)$ must be in $H$ (Property 1)

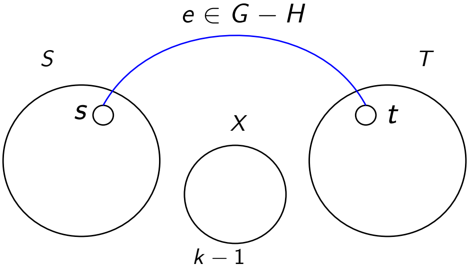



**Lemma (Property 1)**

*Every edge whose endpoints are less than $2k$ connected in $G$ exists in $H$ whp.*

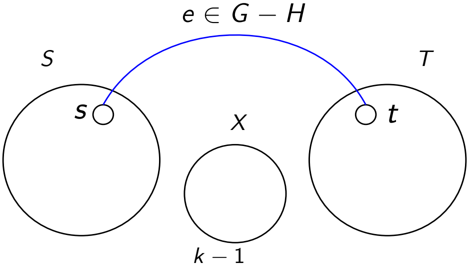
# Correctness

- Case 1: $s$ and $t$ have $< 2k$ vertex-disjoint paths between them in $G$
- The edge $(s,t)$ must be in $H$ (Property 1)



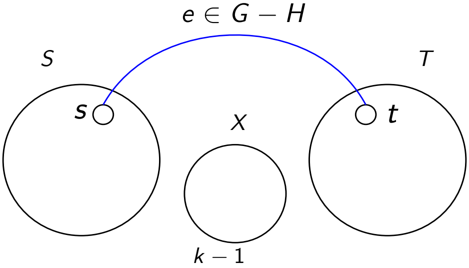
# Correctness

- Case 2: $s, t$ have $\geq 2k$ vertex-disjoint paths between them in $G$



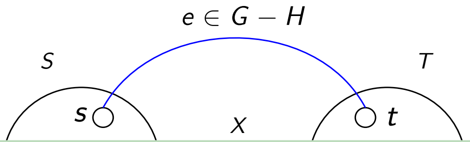
# Correctness

- Case 2: $s, t$ have $\geq 2k$ vertex-disjoint paths between them in $G$
- This means $s, t$ have $\geq k$ vertex-disjoint paths between them in $H$



# Correctness

- Case 2: $s, t$ have $\geq 2k$ vertex-disjoint paths between them in $G$
- This means $s, t$ have $\geq k$ vertex-disjoint paths between them in $H$

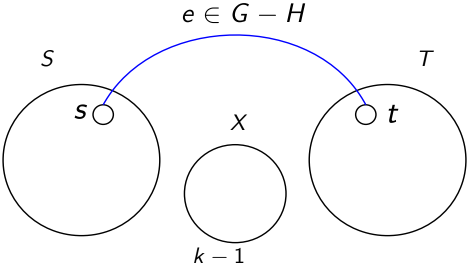



**Lemma (Property 2)**

*Every pair of vertices that is at least $2k$ connected in $G$ is at least $k$ connected in $H$ whp.*

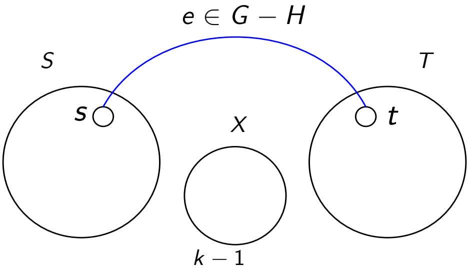
## Correctness

- Case 2: $s, t$ have $\geq 2k$ vertex-disjoint paths between them in $G$
- This means $s, t$ have $\geq k$ vertex-disjoint paths between them in $H$



# Correctness

- Case 2: $s, t$ have $\geq 2k$ vertex-disjoint paths between them in $G$
- This means $s, t$ have $\geq k$ vertex-disjoint paths between them in $H$
- Thus, deleting $k-1$ vertices $(X)$ should not disconnect $s$ and $t$ in $H$



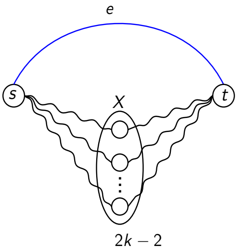
# Property 1

**Lemma**

*Every edge whose endpoints are less than $2k$ connected in $G$ exists in $H$ whp.*

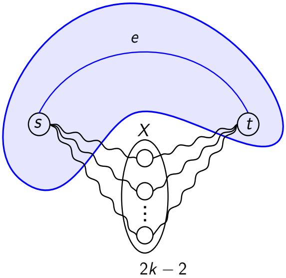
# Property 1

- Consider an edge $(s, t)$ whose endpoints are less than $2k$ connected
- If $s, t$ are sampled and $X$ is not then edge $(s, t)$ is in $H$
- Every spanning forest will contain the edge $(s, t)$



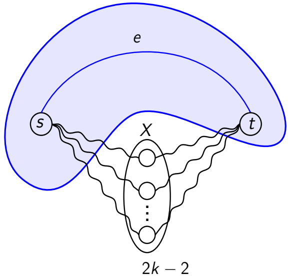
# Property 1

- $\Pr(s \text{ and } t \text{ sampled}) = 1/k^2$



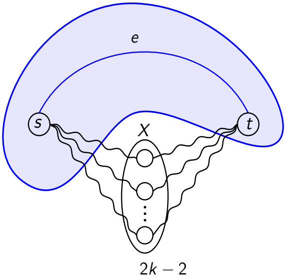
# Property 1

- $\Pr(s \text{ and } t \text{ sampled}) = 1/k^2$
- $\Pr(X \text{ not sampled}) = (1 - 1/k)^{2k-2}$
  $= \Theta(1)$



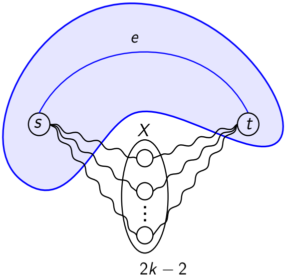
# Property 1

- $\Pr(s \text{ and } t \text{ sampled}) = 1/k^2$
- $\Pr(X \text{ not sampled}) = (1 - 1/k)^{2k-2}$
  $= \Theta(1)$
- $\text{Iterations} = O(k^2 \log n)$



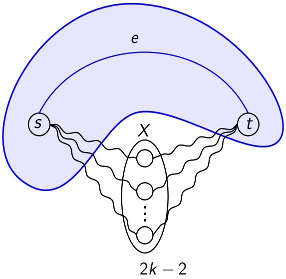
# Property 1

- $\Pr(s \text{ and } t \text{ sampled}) = 1/k^2$
- $\Pr(X \text{ not sampled}) = (1 - 1/k)^{2k-2}$
  $= \Theta(1)$
- $\text{Iterations} = O(k^2 \log n)$
- $\Pr(\text{failure}) = (1 - \Theta(1/k^2))^{O(k^2 \log n)}$
  $\leq 1/\text{poly}(n)$



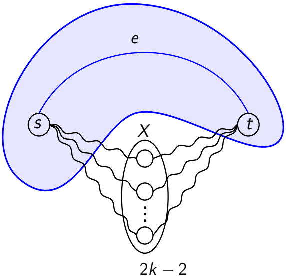
# Property 1

- Thus, whp in some iteration $s, t$ are sampled and $X$ is not



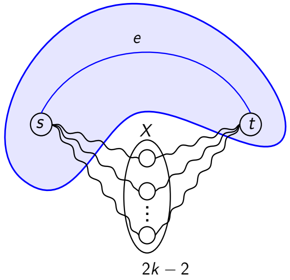
# Property 1

- Thus, whp in some iteration $s, t$ are sampled and $X$ is not
- The spanning tree in this iteration contains the edge $(s, t)$



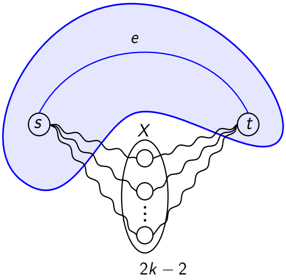
# Property 1

- Thus, whp in some iteration $s, t$ are sampled and $X$ is not
- The spanning tree in this iteration contains the edge $(s, t)$
- Thus, the certificate $H$ contains the edge $(s, t)$



# Property 1

- Thus, whp in some iteration $s, t$ are sampled and $X$ is not
- The spanning tree in this iteration contains the edge $(s, t)$
- Thus, the certificate $H$ contains the edge $(s, t)$
- Union bound over all such pairs
- Property 1 holds whp



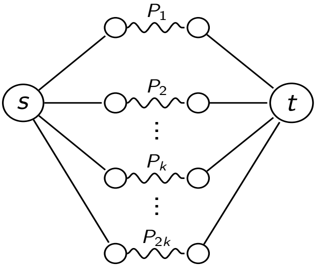
# Property 2

**Lemma**

*Every pair of vertices that is at least $2k$ connected in $G$ is at least $k$ connected in $H$ whp [GMT15].*

# Property 2

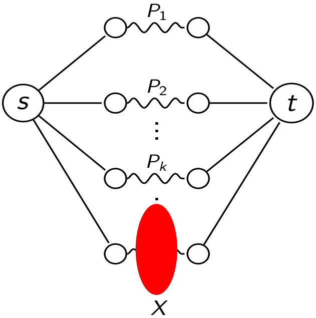
- Consider pair $s, t$ that is at least $2k$ connected



# Property 2

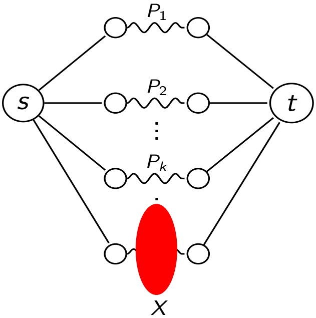
- Consider pair $s, t$ that is at least $2k$ connected

- Consider an arbitrary set $X$ of size $k-1$ (not containing $s, t$)



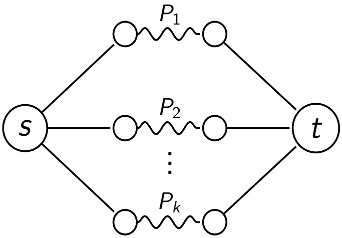
# Property 2

- Consider pair $s, t$ that is at least $2k$ connected
- Consider an arbitrary set $X$ of size $k - 1$ (not containing $s, t$)
- We will show that with very high probability $s, t$ are connected in the certificate $H$ even when $X$ is deleted



# Property 2

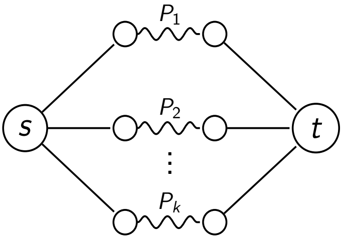
- We only focus on paths $P_1$ to $P_k$



# Property 2

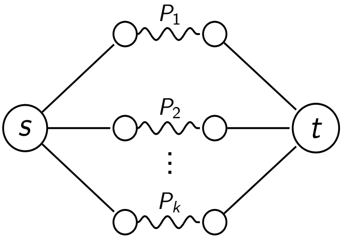
- We only focus on paths $P_1$ to $P_k$

- $\Pr(X \text{ not sampled}) = (1 - 1/k)^{k-1} = \Theta(1)$



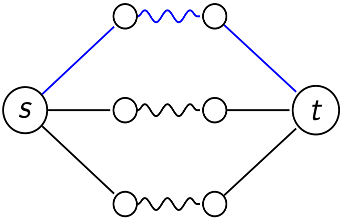
# Property 2

- We only focus on paths $P_1$ to $P_k$
- $\Pr(X \text{ not sampled}) = (1 - 1/k)^{k-1} = \Theta(1)$
- Consider only iterations where $X$ is not sampled (const fraction)



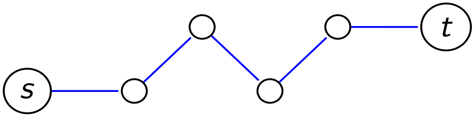
# Spanning Forest

- Need to sample at least one entire path



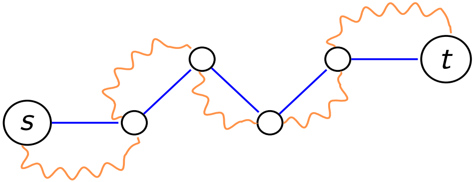
# Spanning Forest

- Need to sample at least one entire path
- Sample each edge in some iteration (i.e. sampling both endpoints)



# Spanning Forest

- Need to sample at least one entire path
- Sample each edge in some iteration (i.e. sampling both endpoints)
- Spanning forest in each iteration will ensure that the endpoints are connected in the certificate $H$



# Property 2

- Consider an edge $e$ on path $P_i$

# Property 2

- Consider an edge $e$ on path $P_i$
- $\Pr(\text{edge } e \text{ sampled}) = 1/k^2$

# Property 2

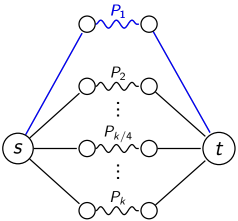
- Consider an edge $e$ on path $P_i$
- $\Pr(\text{edge } e \text{ sampled}) = 1/k^2$
- $\Pr(\text{edge } e \text{ not sampled in any iteration}) = (1 - 1/k^2)^{O(k^2 \log n)}$
  $\leq 1/\mathrm{poly}(n)$

# Property 2

- Consider an edge $e$ on path $P_i$
- $\Pr(\text{edge } e \text{ sampled}) = 1/k^2$
- $\Pr(\text{edge } e \text{ not sampled in any iteration}) = (1 - 1/k^2)^{O(k^2 \log n)}$
  $\leq 1/\mathrm{poly}(n)$
- Union bound over all edges in $P_i$
- An entire $P_i$ is sampled whp

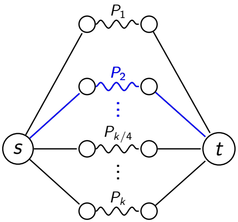
# Property 2

- An entire $P_i$ is sampled whp



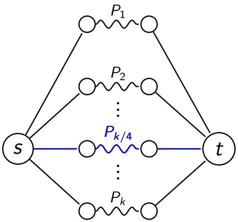
# Property 2

- An entire $P_i$ is sampled whp



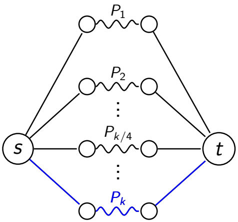
# Property 2

- An entire $P_i$ is sampled whp



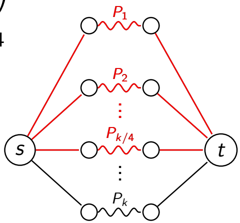
# Property 2

- An entire $P_i$ is sampled whp



# Property 2

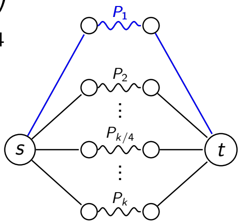
- An entire $P_i$ is sampled whp
- $$\Pr\left(\geq k/4\ P_i\text{'s not sampled}\right) \leq \binom{k}{k/4}\left(\frac{1}{poly(n)}\right)^{k/4}$$
$$\leq 2^k \cdot \left(\frac{1}{poly(n)}\right)^{k/4}$$
$$\leq \left(\frac{1}{poly(n)}\right)^{k}$$



# Property 2

- An entire $P_i$ is sampled whp

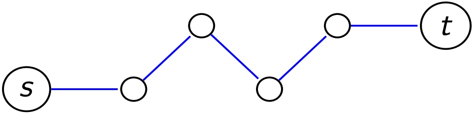
- $$\Pr(\geq k/4\ P_i\text{'s not sampled}) \leq \binom{k}{k/4}\left(\frac{1}{poly(n)}\right)^{k/4} \leq 2^k \cdot \left(\frac{1}{poly(n)}\right)^{k/4} \leq \left(\frac{1}{poly(n)}\right)^{k}$$




- Thus at least one entire path is sampled with very high probability

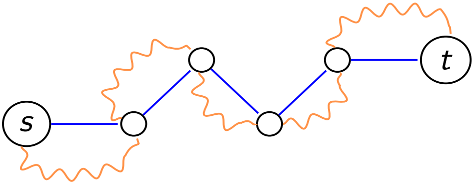
# Property 2

- At least one entire path is sampled with very high probability



# Property 2

- At least one entire path is sampled with very high probability
- $s$ and $t$ are connected in the certificate $H$ with very high probability $(1 - 1/\mathrm{poly}(n^k))$



# Property 2

- At least one entire path is sampled with very high probability

- $s$ and $t$ are connected in the certificate $H$ with very high probability $(1 - 1/\mathrm{poly}(n^k))$

- Union bound over all pairs $s, t$ and sets $X$: $n^2 \cdot n^k$ choices

# Property 2

- At least one entire path is sampled with very high probability
- $s$ and $t$ are connected in the certificate $H$ with very high probability $(1 - 1/\mathrm{poly}(n^k))$
- Union bound over all pairs $s, t$ and sets $X$: $n^2 \cdot n^k$ choices
- Property 2 holds whp

# Summary

We have two key properties:

**Lemma (Property 1)**

*Every edge whose endpoints are less than $2k$ connected in $G$ exists in $H$ whp.*

**Lemma (Property 2)**

*Every pair of vertices that is at least $2k$ connected in $G$ is at least $k$ connected in $H$ whp. [GMT15]*

# Summary

- There is a dynamic streaming algorithm that whp outputs whether the input graph $G$ is $k$-vertex connected or not using $\widetilde{O}(kn)$ space.

# Summary

- There is a dynamic streaming algorithm that whp outputs whether the input graph $G$ is $k$-vertex connected or not using $\widetilde{O}(kn)$ space.
- It does so by outputting a certificate of $k$-vertex connectivity.

# Summary

- There is a dynamic streaming algorithm that whp outputs whether the input graph $G$ is $k$-vertex connected or not using $\widetilde{O}(kn)$ space.
- It does so by outputting a certificate of $k$-vertex connectivity.
- The lower bound of [SW15] is $\Omega(kn)$, making our algorithm optimal (up to polylog factors)

# Summary

- There is a dynamic streaming algorithm that whp outputs whether the input graph $G$ is $k$-vertex connected or not using $\widetilde{O}(kn)$ space.
- It does so by outputting a certificate of $k$-vertex connectivity.
- The lower bound of [SW15] is $\Omega(kn)$, making our algorithm optimal (up to polylog factors)
- The lower bound also holds when outputting a certificate is not required

# Summary

- There is a dynamic streaming algorithm that whp outputs whether the input graph $G$ is $k$-vertex connected or not using $\widetilde{O}(kn)$ space.
- It does so by outputting a certificate of $k$-vertex connectivity.
- The lower bound of [SW15] is $\Omega(kn)$, making our algorithm optimal (up to polylog factors)
- The lower bound also holds when outputting a certificate is not required
- We extend this lower bound to multiple passes and give a lower bound of $\Omega(kn/p)$ for $p$-pass insertion-only streaming algorithms.

# Open Problems

- We have settled the space of the $k$-vertex connectivity problem only up to polylog factors. So the question of optimal space bounds (up to constant factors) is still open.

- Our lower bound and those of Sun and Woodruff [SW15] use duplicate edges. Obtaining lower bounds for simple graphs is an open problem.

# Open Problems

- We have settled the space of the $k$-vertex connectivity problem only up to polylog factors. So the question of optimal space bounds (up to constant factors) is still open.

- Our lower bound and those of Sun and Woodruff [SW15] use duplicate edges. Obtaining lower bounds for simple graphs is an open problem.

**Thank you!**

# References I


Sepehr Assadi, Yu Chen, and Sanjeev Khanna, *Sublinear algorithms for $(\Delta + 1)$ vertex coloring*, Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019, 2019, pp. 767–786.

Kook Jin Ahn, Sudipto Guha, and Andrew McGregor, *Analyzing graph structure via linear measurements*, Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012, 2012, pp. 459–467.

———, *Graph sketches: sparsification, spanners, and subgraphs*, Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012, 2012, pp. 5–14.

# References II


- Li Chen, Rasmus Kyng, Yang P Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva, *Maximum flow and minimum-cost flow in almost-linear time*, arXiv preprint arXiv:2203.00671. To appear in FOCS 2022. (2022).
- Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang, *On graph problems in a semi-streaming model*, Theor. Comput. Sci. **348** (2005), no. 2-3, 207–216.
- Sudipto Guha, Andrew McGregor, and David Tench, *Vertex and hyperedge connectivity in dynamic graph streams*, Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015, 2015, pp. 241–247.

# References III

Jason Li, Danupon Nanongkai, Debmalya Panigrahi, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai, *Vertex connectivity in poly-logarithmic max-flows*, Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, 2021, pp. 317–329.

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