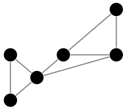
Tight Bounds for Vertex Connectivity in Dynamic Streams

Sepehr Assadi & Vihan Shah

Rutgers University

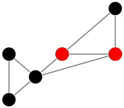
Vertex Connectivity

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



Vertex Connectivity

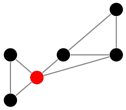
- Undirected Graph $G = (V, E)$
- Vertex Connectivity: Minimum number of vertices that need to be deleted to disconnect G



2

Vertex Connectivity

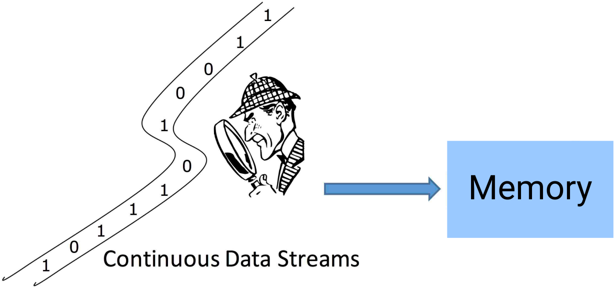
- Undirected Graph $G = (V, E)$
- Vertex Connectivity: Minimum number of vertices that need to be deleted to disconnect G



1

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)

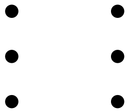


Streaming Models

Insertion-Only

Streaming Models

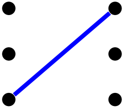
Insertion-Only



Streaming Models

Insertion-Only

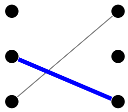
e_1



Streaming Models

Insertion-Only

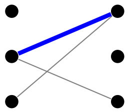
e_1	e_2



Streaming Models

Insertion-Only

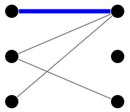
e_1	e_2	e_3



Streaming Models

Insertion-Only

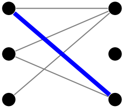
e_1	e_2	e_3	e_4



Streaming Models

Insertion-Only

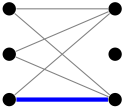
e_1	e_2	e_3	e_4	e_5



Streaming Models

Insertion-Only

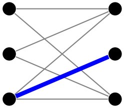
e_1	e_2	e_3	e_4	e_5	e_6



Streaming Models

Insertion-Only

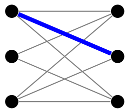
e_1	e_2	e_3	e_4	e_5	e_6	e_7



Streaming Models

Insertion-Only

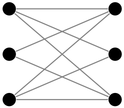
e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Streaming Models

Insertion-Only (finite stream)

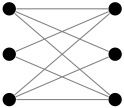
e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Streaming Models

Insertion-Only (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8

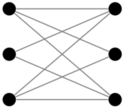


Dynamic

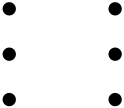
Streaming Models

Insertion-Only (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



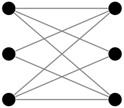
Dynamic



Streaming Models

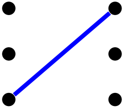
Insertion-Only (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Dynamic

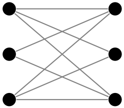
e_1



Streaming Models

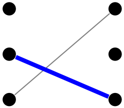
Insertion-Only (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Dynamic

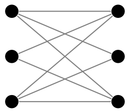
e_1	e_2



Streaming Models

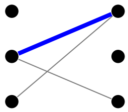
Insertion-Only (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Dynamic

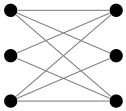
e_1	e_2	e_3



Streaming Models

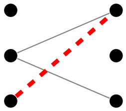
Insertion-Only (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Dynamic

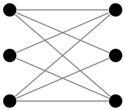
e_1	e_2	e_3	$\overline{e_1}$



Streaming Models

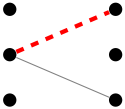
Insertion-Only (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Dynamic

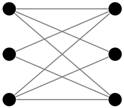
e_1	e_2	e_3	$\overline{e_1}$	$\overline{e_3}$



Streaming Models

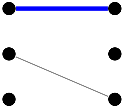
Insertion-Only (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Dynamic

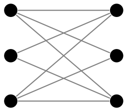
e_1	e_2	e_3	$\overline{e_1}$	$\overline{e_3}$	e_4



Streaming Models

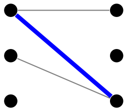
Insertion-Only (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Dynamic

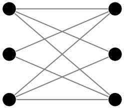
e_1	e_2	e_3	$\overline{e_1}$	$\overline{e_3}$	e_4	e_5



Streaming Models

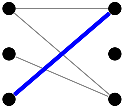
Insertion-Only (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Dynamic

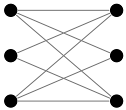
e_1	e_2	e_3	$\overline{e_1}$	$\overline{e_3}$	e_4	e_5	e_1



Streaming Models

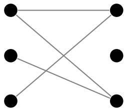
Insertion-Only (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Dynamic (finite stream)

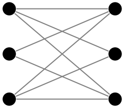
e_1	e_2	e_3	$\overline{e_1}$	$\overline{e_3}$	e_4	e_5	e_1



Streaming Models

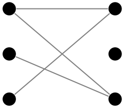
Insertion-Only (finite stream)

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]

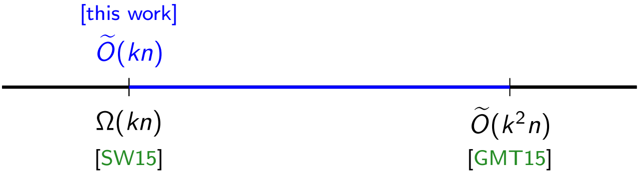
There is a gap of factor k between the best known upper and lower bound in dynamic streams

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.

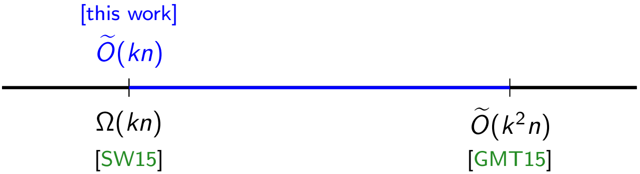


Our Results

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 k-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])

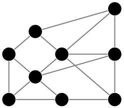
The upper bound also works for multi-graphs

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

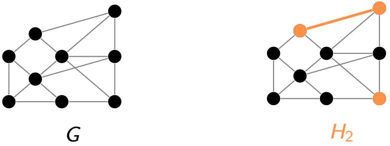
G H_1

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

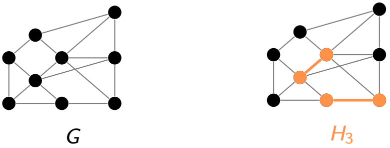


Algorithm of [GMT15]

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

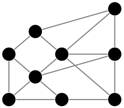


Algorithm of [GMT15]

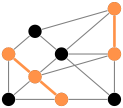
For $i = 1$ to $r = O(k^2 \log n)$:

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



H_4

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

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.

You can visit my poster!

Thank you!

References I

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.

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.