Parameterized Streaming Algorithms

Rajesh Chitnis

Workshop on Data Summarization
22nd March 2018

Parameterized Streaming Algorithms

Rajesh Chitnis

Workshop on Data Summarization
22nd March 2018

THE UNIVERSITY OF WARWICK

Outline of Talk

Outline of Talk

- Streaming Algorithms

Outline of Talk

- Streaming Algorithms
- Parameterized Algorithms

Outline of Talk

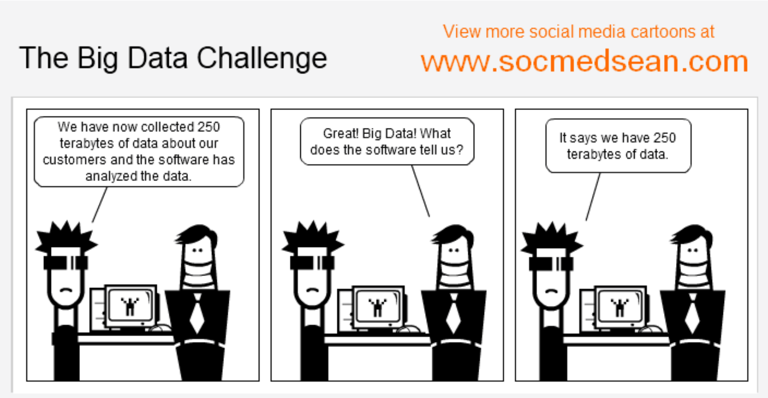
- Streaming Algorithms
- Parameterized Algorithms
- Parameterized Streaming Algorithms

Outline of Talk

- Streaming Algorithms
- Parameterized Algorithms
- Parameterized Streaming Algorithms

Streaming algorithms

BIG Data

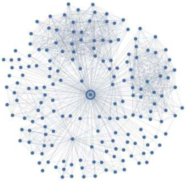


Streaming algorithms
BIG graphs

Streaming algorithms
BIG graphs

Social networks: Google+,Facebook and Twitter

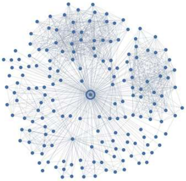
- 10^9 nodes



Streaming algorithms
BIG graphs

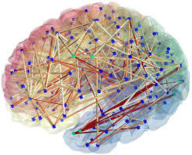
Social networks: Google+,Facebook and Twitter

- 10^9 nodes



Biological networks: Brain connectome

- 10^9 nodes



Streaming algorithms
BIG graphs

Social networks: Google+,Facebook and Twitter

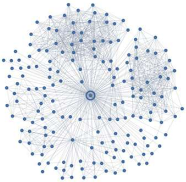
- 10^9 nodes

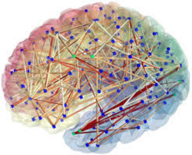
Biological networks: Brain connectome

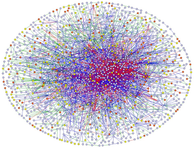
- 10^9 nodes

Computer networks: Web graph

- 2^{32} nodes



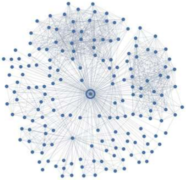




Streaming algorithms
BIG graphs

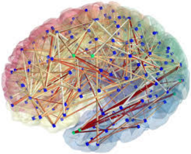
Social networks: Google+,Facebook and Twitter

- 10^9 nodes



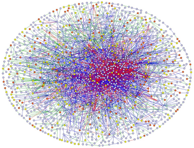
Biological networks: Brain connectome

- 10^9 nodes



Computer networks: Web graph

- 2^{32} nodes



Road networks: USA map in Google Maps

- 10^8 intersection nodes

Streaming algorithms

... on graphs

- Model

Streaming algorithms

... on graphs

- Model
  - Vertex set V is known
  - Edges arrive one-by-one
  - Cannot store all the edges

Streaming algorithms

... on graphs

- Model
  - Vertex set V is known
  - Edges arrive one-by-one
  - Cannot store all the edges
- Cannot control which order edges arrive in
  - More general model also allows edges to be deleted

Streaming algorithms

... on graphs

- Model
  - Vertex set V is known
  - Edges arrive one-by-one
  - Cannot store all the edges
- Cannot control which order edges arrive in
  - More general model also allows edges to be deleted
- Still want to solve our favorite problems

Streaming algorithms

... on graphs

- Model
  - Vertex set V is known
  - Edges arrive one-by-one
  - Cannot store all the edges
- Cannot control which order edges arrive in
  - More general model also allows edges to be deleted
- Still want to solve our favorite problems
  - Max Matching (MM)
  - Min Vertex Cover (VC)
  - ......

Streaming algorithms

... on graphs

- Model
  - Vertex set V is known
  - Edges arrive one-by-one
  - Cannot store all the edges
- Cannot control which order edges arrive in
  - More general model also allows edges to be deleted
- Still want to solve our favorite problems
  - Max Matching (MM)
  - Min Vertex Cover (VC)
  - ......
- Easy upper bound for space is $O(n^2)$
- Finding a min vertex cover has $\Omega(n^2)$ lower bound

Streaming algorithms

... on graphs

- Model
  - Vertex set V is known
  - Edges arrive one-by-one
  - Cannot store all the edges
- Cannot control which order edges arrive in
  - More general model also allows edges to be deleted
- Still want to solve our favorite problems
  - Max Matching (MM)
  - Min Vertex Cover (VC)
  - ......
- Easy upper bound for space is $O(n^2)$
- Finding a min vertex cover has $\Omega(n^2)$ lower bound
  - Reduction from INDEX
  - Essentially need to have stored all edges

Outline of Talk

- Streaming Algorithms
- Parameterized Algorithms
- Parameterized Streaming Algorithms

Why, and what are parameterized algorithms?

Potential drawback of Classical Complexity?

- Classical complexity measures the running time of an algorithm as a function of the input size alone.

Why, and what are parameterized algorithms?

Potential drawback of Classical Complexity?

- Classical complexity measures the running time of an algorithm as a function of the input size alone.
  - Maximum Matching can be solved in $O(m\sqrt{n})$ time

Why, and what are parameterized algorithms?

Potential drawback of Classical Complexity?

- Classical complexity measures the running time of an algorithm as a function of the input size alone.
    - Maximum Matching can be solved in $O(m\sqrt{n})$ time
- Consider the problems of INDEPENDENT SET and VERTEX COVER.

Why, and what are parameterized algorithms?

Potential drawback of Classical Complexity?

- Classical complexity measures the running time of an algorithm as a function of the input size alone.
  - Maximum Matching can be solved in $O(m\sqrt{n})$ time
- Consider the problems of Independent Set and Vertex Cover.

Independent Set
Input: An undirected graph $G = (V, E)$
Output : Find a set $S \subseteq V$ of maximum size such that no two vertices of S form an edge.

Vertex Cover
Input: An undirected graph $G = (V, E)$
Output : Find a set $X \subseteq V$ of minimum size such that X intersects every edge.

Why, and what are parameterized algorithms?

Potential drawback of Classical Complexity?

- Classical complexity measures the running time of an algorithm as a function of the input size alone.
  - Maximum Matching can be solved in $O(m\sqrt{n})$ time
- Consider the problems of Independent Set and Vertex Cover.

> Independent Set
> Input: An undirected graph $G = (V, E)$
> Output : Find a set $S \subseteq V$ of maximum size such that no two vertices of S form an edge.

> Vertex Cover
> Input: An undirected graph $G = (V, E)$
> Output : Find a set $X \subseteq V$ of minimum size such that X intersects every edge.

- S is an independent set if and only if $V \setminus S$ is a vertex cover

Why, and what are parameterized algorithms?

Potential drawback of Classical Complexity?

- Classical complexity measures the running time of an algorithm as a function of the input size alone.
  - Maximum Matching can be solved in $O(m\sqrt{n})$ time
- Consider the problems of Independent Set and Vertex Cover.

Independent Set
Input: An undirected graph $G = (V, E)$
Output : Find a set $S \subseteq V$ of maximum size such that no two vertices of S form an edge.

Vertex Cover
Input: An undirected graph $G = (V, E)$
Output : Find a set $X \subseteq V$ of minimum size such that X intersects every edge.

- S is an independent set if and only if $V \setminus S$ is a vertex cover
- Hence, the classical complexity of Independent Set and Vertex Cover is the same!

Why, and what are parameterized algorithms?

Potential drawback of Classical Complexity?

- Classical complexity measures the running time of an algorithm as a function of the input size alone.
  - Maximum Matching can be solved in $O(m\sqrt{n})$ time
- Consider the problems of INDEPENDENT SET and VERTEX COVER.

INDEPENDENT SET
Input: An undirected graph $G = (V, E)$
Output : Find a set $S \subseteq V$ of maximum size such that no two vertices of S form an edge.

VERTEX COVER
Input: An undirected graph $G = (V, E)$
Output : Find a set $X \subseteq V$ of minimum size such that X intersects every edge.

- S is an independent set if and only if $V \setminus S$ is a vertex cover
- Hence, the classical complexity of INDEPENDENT SET and VERTEX COVER is the same!
  - Any $f(n)$ algorithm for one problem also works for the other.

Why, and what are parameterized algorithms?

Adding a parameter

- In the classical VERTEX COVER problem, the goal is to find a minimum independent set.

Why, and what are parameterized algorithms?

Adding a parameter

- In the classical Vertex Cover problem, the goal is to find a minimum independent set.
- In the parameterized Vertex Cover problem, given a parameter k, we only want to know if G has a vertex cover of size at most k or not.

Why, and what are parameterized algorithms?

Adding a parameter

- In the classical Vertex Cover problem, the goal is to find a minimum independent set.
- In the parameterized Vertex Cover problem, given a parameter k, we only want to know if G has a vertex cover of size at most k or not.
- The goal is to develop fast algorithms when k is small, even if the input size n is large.

Why, and what are parameterized algorithms?

Adding a parameter

- In the classical VERTEX COVER problem, the goal is to find a minimum independent set.
- In the parameterized VERTEX COVER problem, given a parameter k, we only want to know if G has a vertex cover of size at most k or not.
- The goal is to develop fast algorithms when k is small, even if the input size n is large.

Definition: A parameterized problem with parameter k and input size n is said to be fixed-parameter tractable (FPT) if it can be solved in time $f(k) \cdot n^{O(1)}$, for some function f.

Why, and what are parameterized algorithms?

Parameterized VERTEX COVER vs INDEPENDENT SET

k-VERTEX COVER
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

k-INDEPENDENT SET
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $S \subseteq V$ of size $\geq k$ such that no two vertices of S form an edge.

Why, and what are parameterized algorithms?

Parameterized VERTEX COVER vs INDEPENDENT SET

k-VERTEX COVER
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

k-INDEPENDENT SET
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $S \subseteq V$ of size $\geq k$ such that no two vertices of S form an edge.

- Pick any edge uv, and branch on choosing either u or v

Why, and what are parameterized algorithms?

Parameterized VERTEX COVER vs INDEPENDENT SET

k-VERTEX COVER
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

k-INDEPENDENT SET
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $S \subseteq V$ of size $\geq k$ such that no two vertices of S form an edge.

- Pick any edge uv, and branch on choosing either u or v
- Binary search tree of depth k

Why, and what are parameterized algorithms?

Parameterized Vertex Cover vs Independent Set

k-Vertex Cover
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

k-Independent Set
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $S \subseteq V$ of size $\geq k$ such that no two vertices of S form an edge.

- Pick any edge uv, and branch on choosing either u or v
- Binary search tree of depth k
- $2^k \cdot n^{O(1)}$ algorithm

Why, and what are parameterized algorithms?

Parameterized Vertex Cover vs Independent Set

k-Vertex Cover
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

- Pick any edge uv, and branch on choosing either u or v
- Binary search tree of depth k
- $2^k \cdot n^{O(1)}$ algorithm

k-Independent Set
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $S \subseteq V$ of size $\geq k$ such that no two vertices of S form an edge.

- $\binom{n}{k} = n^{O(k)}$ is trivial

Why, and what are parameterized algorithms?

Parameterized Vertex Cover vs Independent Set

k-Vertex Cover
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

- Pick any edge uv, and branch on choosing either u or v
- Binary search tree of depth k
- $2^k \cdot n^{O(1)}$ algorithm

k-Independent Set
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $S \subseteq V$ of size $\geq k$ such that no two vertices of S form an edge.

- $\binom{n}{k} = n^{O(k)}$ is trivial
- No $f(k) \cdot n^{o(k)}$ algorithm for any f (under ETH)

Why, and what are parameterized algorithms?

Parameterized Vertex Cover vs Independent Set

k-Vertex Cover
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

- Pick any edge uv, and branch on choosing either u or v
- Binary search tree of depth k
- $2^k \cdot n^{O(1)}$ algorithm

k-Independent Set
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $S \subseteq V$ of size $\geq k$ such that no two vertices of S form an edge.

- $\binom{n}{k} = n^{O(k)}$ is trivial
- No $f(k) \cdot n^{o(k)}$ algorithm for any f (under ETH)

- Thus, Vertex Cover and Independent Set are very different with respect to parameterized complexity

Why, and what are parameterized algorithms?

Parameterized Vertex Cover vs Independent Set

k-Vertex Cover
Input: An undirected graph $G = (V, E)$ Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

- Pick any edge uv, and branch on choosing either u or v
- Binary search tree of depth k
- $2^k \cdot n^{O(1)}$ algorithm

k-Independent Set
Input: An undirected graph $G = (V, E)$ Output : Does there exist a set $S \subseteq V$ of size $\geq k$ such that no two vertices of S form an edge.

- $\binom{n}{k} = n^{O(k)}$ is trivial
- No $f(k) \cdot n^{o(k)}$ algorithm for any f (under ETH)

- Thus, Vertex Cover and Independent Set are very different with respect to parameterized complexity
  - Although they were equivalent with respect to classical complexity

Why, and what are parameterized algorithms?

Parameterized Vertex Cover vs Independent Set

k-Vertex Cover
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

- Pick any edge uv, and branch on choosing either u or v
- Binary search tree of depth k
- $2^k \cdot n^{O(1)}$ algorithm

k-Independent Set
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $S \subseteq V$ of size $\geq k$ such that no two vertices of S form an edge.

- $\binom{n}{k} = n^{O(k)}$ is trivial
- No $f(k) \cdot n^{o(k)}$ algorithm for any f (under ETH)

- Thus, Vertex Cover and Independent Set are very different with respect to parameterized complexity
  - Although they were equivalent with respect to classical complexity
- So this notion of parameterized (time) complexity actually does give us some insight .....

- The complexity classes of parameterized complexity are:

$$\mathsf{FPT} \subseteq \mathsf{W[1]} \subseteq \mathsf{W[2]} \subseteq \ldots \subseteq \mathsf{W[i]} \subseteq \ldots$$

- The complexity classes of parameterized complexity are:

$$\text{FPT} \subseteq \text{W[1]} \subseteq \text{W[2]} \subseteq \ldots \subseteq \text{W[i]} \subseteq \ldots$$

- FPT: Solvable in $f(k) \cdot n^{O(1)}$ time for some function f

Parameterized Algorithms

Complexity Classes

- The complexity classes of parameterized complexity are:

$$\text{FPT} \subseteq \text{W[1]} \subseteq \text{W[2]} \subseteq \ldots \subseteq \text{W[i]} \subseteq \ldots$$

- FPT: Solvable in $f(k) \cdot n^{O(1)}$ time for some function f
  - The "P" of the parameterized world

- The complexity classes of parameterized complexity are:

$$\text{FPT} \subseteq \text{W[1]} \subseteq \text{W[2]} \subseteq \ldots \subseteq \text{W[i]} \subseteq \ldots$$

- FPT: Solvable in $f(k) \cdot n^{O(1)}$ time for some function f
  - The "P" of the parameterized world
- $W[i]$-hard: Do not expect $f(k) \cdot n^{O(1)}$ algorithms for any f

Parameterized Algorithms

Complexity Classes

- The complexity classes of parameterized complexity are:

$$\text{FPT} \subseteq \text{W[1]} \subseteq \text{W[2]} \subseteq \ldots \subseteq \text{W[i]} \subseteq \ldots$$

- FPT: Solvable in $f(k) \cdot n^{O(1)}$ time for some function f
  - The "P" of the parameterized world
- $W[i]$-hard: Do not expect $f(k) \cdot n^{O(1)}$ algorithms for any f
  - The "NP-hard" of the parameterized world

- The complexity classes of parameterized complexity are:

$$\text{FPT} \subseteq \text{W[1]} \subseteq \text{W[2]} \subseteq \ldots \subseteq \text{W[i]} \subseteq \ldots$$

- FPT: Solvable in $f(k) \cdot n^{O(1)}$ time for some function f
  - The "P" of the parameterized world
- $W[i]$-hard: Do not expect $f(k) \cdot n^{O(1)}$ algorithms for any f
  - The "NP-hard" of the parameterized world
  - The classes $W[i]$ have technical definitions which we skip here

- The complexity classes of parameterized complexity are:

$$\text{FPT} \subseteq \text{W[1]} \subseteq \text{W[2]} \subseteq \ldots \subseteq \text{W[i]} \subseteq \ldots$$

- FPT: Solvable in $f(k) \cdot n^{O(1)}$ time for some function f
  - The "P" of the parameterized world
- $W[i]$-hard: Do not expect $f(k) \cdot n^{O(1)}$ algorithms for any f
  - The "NP-hard" of the parameterized world
  - The classes $W[i]$ have technical definitions which we skip here
  - Clique is an example of a W[1]-hard problem

- The complexity classes of parameterized complexity are:

$$\text{FPT} \subseteq \text{W[1]} \subseteq \text{W[2]} \subseteq \ldots \subseteq \text{W[i]} \subseteq \ldots$$

- FPT: Solvable in $f(k) \cdot n^{O(1)}$ time for some function f
  - The "P" of the parameterized world
- $W[i]$-hard: Do not expect $f(k) \cdot n^{O(1)}$ algorithms for any f
  - The "NP-hard" of the parameterized world
  - The classes $W[i]$ have technical definitions which we skip here
  - Clique is an example of a W[1]-hard problem
  - Set Cover is an example of a W[2]-hard problem

Parameterized Algorithms

Kernels

- Different from all the kernels you've heard before in this workshop!

Parameterized Algorithms

Kernels

- Different from all the kernels you've heard before in this workshop!
- Kernel is the small, essential part of the big, hard input

Parameterized Algorithms

Kernels

- Different from all the kernels you've heard before in this workshop!
- Kernel is the small, essential part of the big, hard input

Parameterized Algorithms

Kernels

- Different from all the kernels you've heard before in this workshop!
- Kernel is the small, essential part of the big, hard input

Parameterized Algorithms

Kernels

- Kernelization is a formal way of preprocessing the input graph

Parameterized Algorithms

Kernels

- Kernelization is a formal way of preprocessing the input graph
- Consider an instance (G, k) of k-VC

Parameterized Algorithms

Kernels

- Kernelization is a formal way of preprocessing the input graph
- Consider an instance (G, k) of k-VC
- Can we build a new graph (G', k') in time $n^{O(1)}$ such that

Parameterized Algorithms

Kernels

- Kernelization is a formal way of preprocessing the input graph
- Consider an instance (G, k) of k-VC
- Can we build a new graph (G', k') in time $n^{O(1)}$ such that
  - $|G'| = g(k)$
  - $k' = h(k)$
  - (G, k) and (G', k') are equivalent

Parameterized Algorithms

Kernels

- Kernelization is a formal way of preprocessing the input graph
- Consider an instance (G, k) of k-VC
- Can we build a new graph (G', k') in time $n^{O(1)}$ such that
  - $|G'| = g(k)$
  - $k' = h(k)$
  - (G, k) and (G', k') are equivalent
- Such a graph G' is called as a $g(k)$-sized kernel for k-VC

Parameterized Algorithms

Kernels

- Kernelization is a formal way of preprocessing the input graph
- Consider an instance (G, k) of k-VC
- Can we build a new graph (G', k') in time $n^{O(1)}$ such that
  - $|G'| = g(k)$
  - $k' = h(k)$
  - (G, k) and (G', k') are equivalent
- Such a graph G' is called as a $g(k)$-sized kernel for k-VC
- Observation: Any vertex of deg $> k$ has to be part of every VC of size $\leq k$

Parameterized Algorithms

Kernels

- Kernelization is a formal way of preprocessing the input graph
- Consider an instance (G, k) of k-VC
- Can we build a new graph (G', k') in time $n^{O(1)}$ such that
  - $|G'| = g(k)$
  - $k' = h(k)$
  - (G, k) and (G', k') are equivalent
- Such a graph G' is called as a $g(k)$-sized kernel for k-VC
- Observation: Any vertex of deg $> k$ has to be part of every VC of size $\leq k$
  - Otherwise we need to include all its neighbors into the VC!

Parameterized Algorithms

Kernels

- Kernelization is a formal way of preprocessing the input graph
- Consider an instance (G, k) of k-VC
- Can we build a new graph (G', k') in time $n^{O(1)}$ such that
  - $|G'| = g(k)$
  - $k' = h(k)$
  - (G, k) and (G', k') are equivalent
- Such a graph G' is called as a $g(k)$-sized kernel for k-VC
- Observation: Any vertex of deg $> k$ has to be part of every VC of size $\leq k$
  - Otherwise we need to include all its neighbors into the VC!
- Consider the following kernel"

Parameterized Algorithms

- Kernelization is a formal way of preprocessing the input graph
- Consider an instance (G, k) of k-VC
- Can we build a new graph (G', k') in time $n^{O(1)}$ such that
  - $|G'| = g(k)$
  - $k' = h(k)$
  - (G, k) and (G', k') are equivalent
- Such a graph G' is called as a $g(k)$-sized kernel for k-VC
- Observation: Any vertex of deg $> k$ has to be part of every VC of size $\leq k$
  - Otherwise we need to include all its neighbors into the VC!
- Consider the following kernel"
  - Find a vertex of degree $> k$. Add it to VC, and delete from graph.
  - Reduce k by 1
  - Repeat

Parameterized Algorithms

Kernels

- Kernelization is a formal way of preprocessing the input graph
- Consider an instance (G, k) of k-VC
- Can we build a new graph (G', k') in time $n^{O(1)}$ such that
  - $|G'| = g(k)$
  - $k' = h(k)$
  - (G, k) and (G', k') are equivalent
- Such a graph G' is called as a $g(k)$-sized kernel for k-VC
- Observation: Any vertex of deg $> k$ has to be part of every VC of size $\leq k$
  - Otherwise we need to include all its neighbors into the VC!
- Consider the following kernel"
  - Find a vertex of degree $> k$. Add it to VC, and delete from graph.
  - Reduce k by 1
  - Repeat
- Finally max degree of resulting graph G' becomes $\leq k$

Parameterized Algorithms

- Kernelization is a formal way of preprocessing the input graph
- Consider an instance (G, k) of k-VC
- Can we build a new graph (G', k') in time $n^{O(1)}$ such that
  - $|G'| = g(k)$
  - $k' = h(k)$
  - (G, k) and (G', k') are equivalent
- Such a graph G' is called as a $g(k)$-sized kernel for k-VC
- Observation: Any vertex of deg $> k$ has to be part of every VC of size $\leq k$
  - Otherwise we need to include all its neighbors into the VC!
- Consider the following kernel"
  - Find a vertex of degree $> k$. Add it to VC, and delete from graph.
  - Reduce k by 1
  - Repeat
- Finally max degree of resulting graph G' becomes $\leq k$
- Observation: If $|E'(G)| > k^2$, then original instance (G, k) of k-VC was NO

Parameterized Algorithms

- Kernelization is a formal way of preprocessing the input graph
- Consider an instance (G, k) of k-VC
- Can we build a new graph (G', k') in time $n^{O(1)}$ such that
  - $|G'| = g(k)$
  - $k' = h(k)$
  - (G, k) and (G', k') are equivalent
- Such a graph G' is called as a $g(k)$-sized kernel for k-VC
- Observation: Any vertex of deg $> k$ has to be part of every VC of size $\leq k$
  - Otherwise we need to include all its neighbors into the VC!
- Consider the following kernel"
  - Find a vertex of degree $> k$. Add it to VC, and delete from graph.
  - Reduce k by 1
  - Repeat
- Finally max degree of resulting graph G' becomes $\leq k$
- Observation: If $|E'(G)| > k^2$, then original instance (G, k) of k-VC was NO

Parameterized Algorithms

Kernel $\Leftrightarrow$ FPT

Parameterized Algorithms

Kernel $\Leftrightarrow$ FPT

- Kernel $\Rightarrow$ FPT

- Kernel $\Rightarrow$ FPT
  - Suppose we have $g(k)$-sized kernel

- Kernel $\Rightarrow$ FPT
  - Suppose we have $g(k)$-sized kernel
  - $n^{O(1)} + \exp(g(k)) = r(k) \cdot n^{O(1)}$

Parameterized Algorithms

Kernel $\Leftrightarrow$ FPT

- Kernel $\Rightarrow$ FPT
  - Suppose we have $g(k)$-sized kernel
  - $n^{O(1)} + \exp(g(k)) = r(k) \cdot n^{O(1)}$
- FPT $\Rightarrow$ Kernel

- Kernel $\Rightarrow$ FPT
  - Suppose we have $g(k)$-sized kernel
  - $n^{O(1)} + \exp(g(k)) = r(k) \cdot n^{O(1)}$
- FPT $\Rightarrow$ Kernel
  - Suppose we have $f(k) \cdot n^c$ algorithm

- Kernel $\Rightarrow$ FPT
  - Suppose we have $g(k)$-sized kernel
  - $n^{O(1)} + \exp(g(k)) = r(k) \cdot n^{O(1)}$
- FPT $\Rightarrow$ Kernel
  - Suppose we have $f(k) \cdot n^c$ algorithm
  - Run the algorithm for n^{c+2} time

- Kernel $\Rightarrow$ FPT
  - Suppose we have $g(k)$-sized kernel
  - $n^{O(1)} + \exp(g(k)) = r(k) \cdot n^{O(1)}$
- FPT $\Rightarrow$ Kernel
  - Suppose we have $f(k) \cdot n^c$ algorithm
  - Run the algorithm for n^{c+2} time
  - If it actually terminates, we have trivial kernel

Parameterized Algorithms

Kernel $\Leftrightarrow$ FPT

- Kernel $\Rightarrow$ FPT
  - Suppose we have $g(k)$-sized kernel
  - $n^{O(1)} + \exp(g(k)) = r(k) \cdot n^{O(1)}$
- FPT $\Rightarrow$ Kernel
  - Suppose we have $f(k) \cdot n^c$ algorithm
  - Run the algorithm for n^{c+2} time
  - If it actually terminates, we have trivial kernel
  - Otherwise $f(k) \cdot n^c > n^{c+2} \Rightarrow f(k) > n^2$, and whole graph is $f(k)$-kernel

Parameterized Algorithms

Kernel $\Leftrightarrow$ FPT

- Kernel $\Rightarrow$ FPT
  - Suppose we have $g(k)$-sized kernel
  - $n^{O(1)} + \exp(g(k)) = r(k) \cdot n^{O(1)}$
- FPT $\Rightarrow$ Kernel
  - Suppose we have $f(k) \cdot n^c$ algorithm
  - Run the algorithm for n^{c+2} time
  - If it actually terminates, we have trivial kernel
  - Otherwise $f(k) \cdot n^c > n^{c+2} \Rightarrow f(k) > n^2$, and whole graph is $f(k)$-kernel

Outline of Talk

- Streaming Algorithms
- Parameterized Algorithms
- Parameterized Streaming Algorithms

Parameterized Streaming Algorithms

How about we introduce some parameters?

Parameterized Streaming Algorithms

How about we introduce some parameters?

What if we try to design streaming algorithms for the parameterized versions of the problem, where the space is a function of both n and k (the solution size)?

Parameterized Streaming Algorithms

How about we introduce some parameters?

What if we try to design streaming algorithms for the parameterized versions of the problem, where the space is a function of both n and k (the solution size)?

k-Vertex Cover
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

Parameterized Streaming Algorithms

How about we introduce some parameters?

What if we try to design streaming algorithms for the parameterized versions of the problem, where the space is a function of both n and k (the solution size)?

k-VERTEX COVER
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

- Space requirement?

Parameterized Streaming Algorithms

How about we introduce some parameters?

What if we try to design streaming algorithms for the parameterized versions of the problem, where the space is a function of both n and k (the solution size)?

k-VERTEX COVER
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

- Space requirement?
  - $f(k)$

Parameterized Streaming Algorithms

How about we introduce some parameters?

What if we try to design streaming algorithms for the parameterized versions of the problem, where the space is a function of both n and k (the solution size)?

k-VERTEX COVER
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

- Space requirement?
  - $f(k)$
  - $f(k) \cdot \text{poly} \log n$

Parameterized Streaming Algorithms

How about we introduce some parameters?

What if we try to design streaming algorithms for the parameterized versions of the problem, where the space is a function of both n and k (the solution size)?

k-Vertex Cover
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

- Space requirement?
  - $f(k)$
  - $f(k) \cdot \text{poly}\log n$
  - $f(k) \cdot \sqrt{n}$

Parameterized Streaming Algorithms

How about we introduce some parameters?

What if we try to design streaming algorithms for the parameterized versions of the problem, where the space is a function of both n and k (the solution size)?

k-Vertex Cover
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

- Space requirement?
  - $f(k)$
  - $f(k) \cdot \text{poly} \log n$
  - $f(k) \cdot \sqrt{n}$
  - $f(k) \cdot n$

Parameterized Streaming Algorithms

How about we introduce some parameters?

What if we try to design streaming algorithms for the parameterized versions of the problem, where the space is a function of both n and k (the solution size)?

k-VERTEX COVER
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

- Space requirement?
  - $f(k)$
  - $f(k) \cdot \text{poly}\log n$
  - $f(k) \cdot \sqrt{n}$
  - $f(k) \cdot n$
  - $f(k) \cdot n \cdot \text{poly}\log n$

Parameterized Streaming Algorithms

How about we introduce some parameters?

What if we try to design streaming algorithms for the parameterized versions of the problem, where the space is a function of both n and k (the solution size)?

k-VERTEX COVER
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

- Space requirement?
  - $f(k)$
  - $f(k) \cdot \text{poly} \log n$
  - $f(k) \cdot \sqrt{n}$
  - $f(k) \cdot n$
  - $f(k) \cdot n \cdot \text{poly} \log n$
  - $O(n^2)$

Parameterized Streaming Algorithms

How about we introduce some parameters?

What if we try to design streaming algorithms for the parameterized versions of the problem, where the space is a function of both n and k (the solution size)?

k-Vertex Cover
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

- Space requirement?
  - $f(k)$
  - $f(k) \cdot \text{poly} \log n$
  - $f(k) \cdot \sqrt{n}$
  - $f(k) \cdot n$
  - $f(k) \cdot n \cdot \text{poly} \log n$
  - $O(n^2)$

- Play same "game" as before, but for space now instead of time!

Parameterized Streaming Algorithms

How about we introduce some parameters?

What if we try to design streaming algorithms for the parameterized versions of the problem, where the space is a function of both n and k (the solution size)?

k-VERTEX COVER
Input: An undirected graph $G = (V, E)$
Output : Does there exist a set $X \subseteq V$ of size $\leq k$ such that X intersects every edge.

- Space requirement?
  - $f(k)$
  - $f(k) \cdot \text{poly} \log n$
  - $f(k) \cdot \sqrt{n}$
  - $f(k) \cdot n$
  - $f(k) \cdot n \cdot \text{poly} \log n$
  - $O(n^2)$

- Play same "game" as before, but for space now instead of time!
- Maybe implement kernels in streaming model?

Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Maintain a maximal matching M
  - Let the vertices of the matching be V_M

Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$

Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$
  - Keep upto k neighbors (and corresponding edges)

Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiagghayi, Monemizadeh ['15]

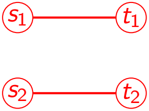
- Maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$
  - Keep upto k neighbors (and corresponding edges)

Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$
  - Keep upto k neighbors (and corresponding edges)

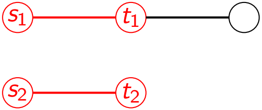


Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$
  - Keep upto k neighbors (and corresponding edges)

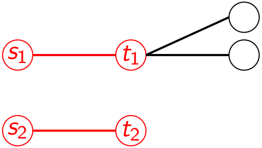


Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$
  - Keep upto k neighbors (and corresponding edges)

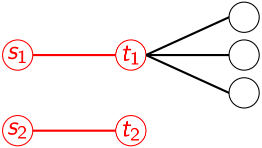


Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$
  - Keep upto k neighbors (and corresponding edges)

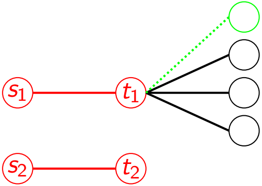


Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$
  - Keep upto k neighbors (and corresponding edges)

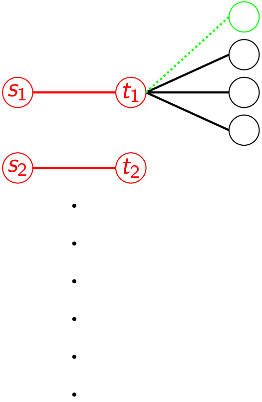


Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$
  - Keep upto k neighbors (and corresponding edges)

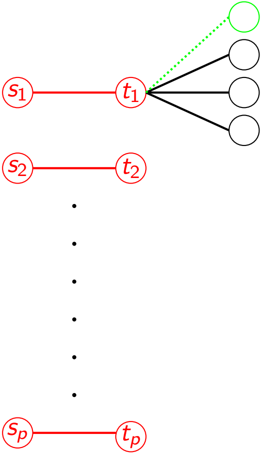


Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$
  - Keep upto k neighbors (and corresponding edges)

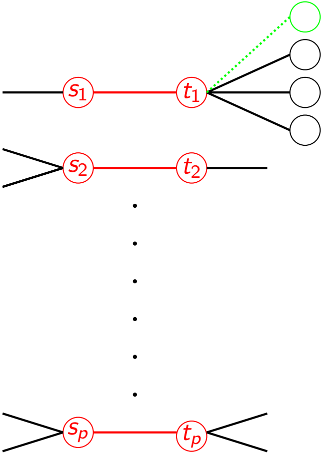


Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$
  - Keep upto k neighbors (and corresponding edges)

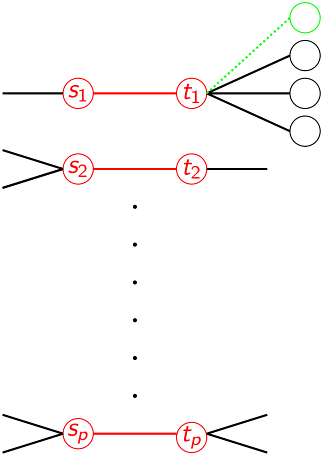


Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Greedily maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$
  - Keep upto k neighbors (and corresponding edges)

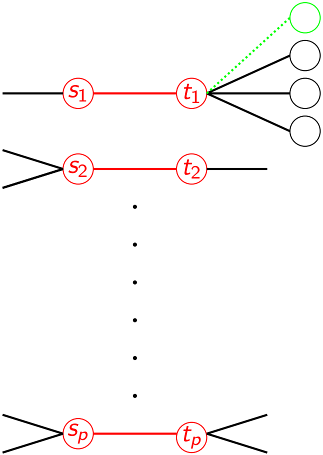


Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Greedily maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$
  - Keep upto k neighbors (and corresponding edges)
- If $p > k$, say NO
  - Hence $p \leq k$

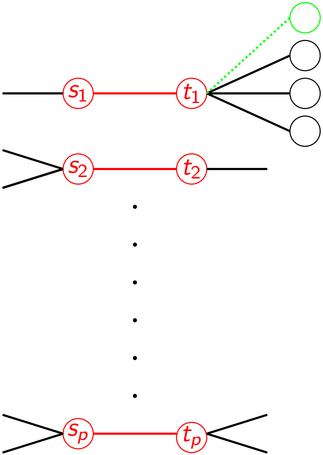


Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Greedily maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$
  - Keep upto k neighbors (and corresponding edges)
- If $p > k$, say NO
  - Hence $p \leq k$
- Let G_M be the graph that we store

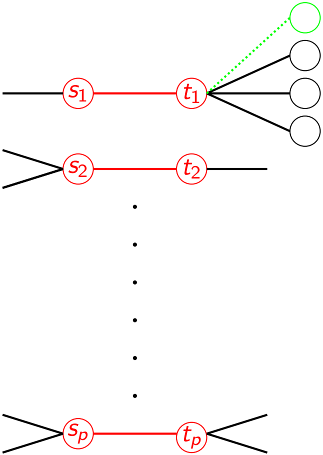


Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Greedily maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$
  - Keep upto k neighbors (and corresponding edges)
- If $p > k$, say NO
  - Hence $p \leq k$
- Let G_M be the graph that we store
  - Everything except green edges

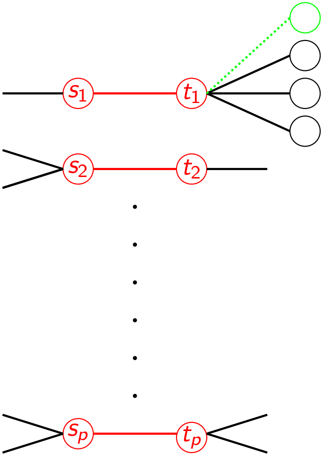


Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Greedily maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$
  - Keep upto k neighbors (and corresponding edges)
- If $p > k$, say NO
  - Hence $p \leq k$
- Let G_M be the graph that we store
  - Everything except green edges
- Lemma: $\text{VC}(G) \leq k \Leftrightarrow \text{VC}(G_M) \leq k$

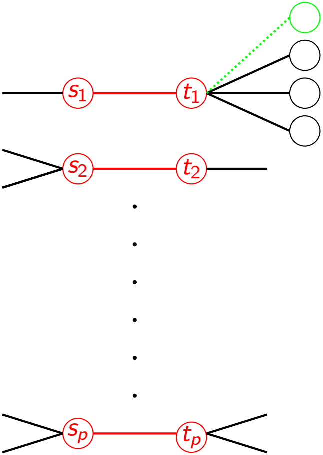


Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Greedily maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$
  - Keep upto k neighbors (and corresponding edges)
- If $p > k$, say NO
  - Hence $p \leq k$
- Let G_M be the graph that we store
  - Everything except green edges
- <u>Lemma</u>: $\mathrm{VC}(G) \leq k \Leftrightarrow \mathrm{VC}(G_M) \leq k$
  - Hence, it is safe to only store the smaller graph G_M

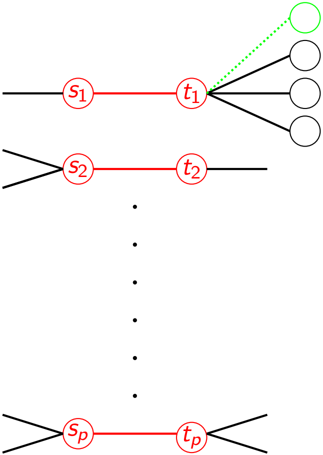


Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Greedily maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$
  - Keep upto k neighbors (and corresponding edges)
- If $p > k$, say NO
  - Hence $p \leq k$
- Let G_M be the graph that we store
  - Everything except green edges
- Lemma: $\text{VC}(G) \leq k \Leftrightarrow \text{VC}(G_M) \leq k$
  - Hence, it is safe to only store the smaller graph G_M
- Idea: Any vertex of degree $> k$ must be in every VC of size $\leq k$; otherwise we need to choose all its neighbors in the VC

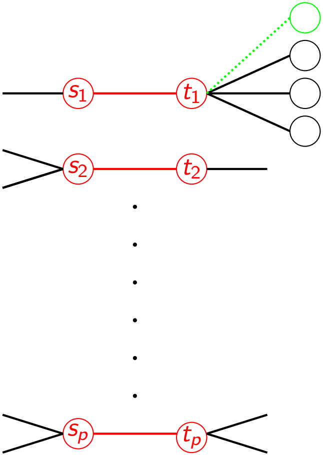


Parameterized Streaming Algorithms

$O(k^2)$ space algorithm for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh ['15]

- Greedily maintain a maximal matching M
  - Let the vertices of the matching be V_M
- For every $x \in V_M$
  - Keep upto k neighbors (and corresponding edges)
- If $p > k$, say NO
  - Hence $p \leq k$
- Let G_M be the graph that we store
  - Everything except green edges
- Lemma: $\text{VC}(G) \leq k \Leftrightarrow \text{VC}(G_M) \leq k$
  - Hence, it is safe to only store the smaller graph G_M
- Idea: Any vertex of degree $> k$ must be in every VC of size $\leq k$; otherwise we need to choose all its neighbors in the VC



Space required is $2p \cdot (k+1) = O(k^2)$ vertices and edges

Parameterized Streaming Algorithms

$\Omega(k^2)$ lower bound for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh [2015]

Parameterized Streaming Algorithms

$\Omega(k^2)$ lower bound for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh [2015]

- INDEX problem

Parameterized Streaming Algorithms

$\Omega(k^2)$ lower bound for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh [2015]

- INDEX problem
  - Alice has $X = (X_1, X_2, \ldots, X_N) \in \{0, 1\}^N$

Parameterized Streaming Algorithms

$\Omega(k^2)$ lower bound for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh [2015]

- INDEX problem
  - Alice has $X = (X_1, X_2, \ldots, X_N) \in \{0,1\}^N$
  - Bob has index $i \in [N]$, and wants to find X_i

Parameterized Streaming Algorithms

$\Omega(k^2)$ lower bound for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh [2015]

- INDEX problem
  - Alice has $X = (X_1, X_2, \ldots, X_N) \in \{0,1\}^N$
  - Bob has index $i \in [N]$, and wants to find X_i
  - Lower bound of $\Omega(N)$ bits

Parameterized Streaming Algorithms

$\Omega(k^2)$ lower bound for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh [2015]

- INDEX problem
  - Alice has $X = (X_1, X_2, \ldots, X_N) \in \{0,1\}^N$
  - Bob has index $i \in [N]$, and wants to find X_i
  - Lower bound of $\Omega(N)$ bits
- Set $k = \sqrt{N}$

Parameterized Streaming Algorithms

$\Omega(k^2)$ lower bound for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh [2015]

- INDEX problem
  - Alice has $X = (X_1, X_2, \ldots, X_N) \in \{0,1\}^N$
  - Bob has index $i \in [N]$, and wants to find X_i
  - Lower bound of $\Omega(N)$ bits
- Set $k = \sqrt{N}$
- Fix a bijection $[k] \times [k] \to [N]$

Parameterized Streaming Algorithms

$\Omega(k^2)$ lower bound for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh [2015]

- INDEX problem
  - Alice has $X = (X_1, X_2, \ldots, X_N) \in \{0,1\}^N$
  - Bob has index $i \in [N]$, and wants to find X_i
  - Lower bound of $\Omega(N)$ bits
- Set $k = \sqrt{N}$
- Fix a bijection $[k] \times [k] \to [N]$
- Introduce $2k$ vertices
  - $v_1, v_2, \ldots, v_k$
  - $w_1, w_2, \ldots, w_k$
- For each $(i,j) \in [k] \times [k]$
  - Alice adds an edge $v_i - w_j$ iff $X_{i,j} = 1$
- Let Bob's index be (i^*, j^*)
- For each $(i,j) \in [k] \times [k]$ such that $i \neq i^*$ and $j \neq j^*$
  - Bob adds two leaves each to v_i and w_j

$\mathrm{VC}(G) = 2k - 2$ if and only if $X_{i^*,j^*} = 0$

Parameterized Streaming Algorithms

$\Omega(k^2)$ lower bound for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh [2015]

- INDEX problem
  - Alice has $X = (X_1, X_2, \ldots, X_N) \in \{0,1\}^N$
  - Bob has index $i \in [N]$, and wants to find X_i
  - Lower bound of $\Omega(N)$ bits
- Set $k = \sqrt{N}$
- Fix a bijection $[k] \times [k] \to [N]$
- Introduce $2k$ vertices
  - $v_1, v_2, \ldots, v_k$
  - $w_1, w_2, \ldots, w_k$
- For each $(i,j) \in [k] \times [k]$
  - Alice adds an edge $v_i - w_j$ iff $X_{i,j} = 1$
- Let Bob's index be (i^*, j^*)
- For each $(i,j) \in [k] \times [k]$ such that $i \neq i^*$ and $j \neq j^*$
  - Bob adds two leaves each to v_i and w_j

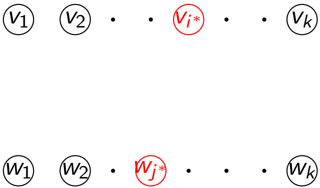
VC(G) $= 2k - 2$ if and only if $X_{i^*,j^*} = 0$

Parameterized Streaming Algorithms

$\Omega(k^2)$ lower bound for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh [2015]

- INDEX problem
  - Alice has $X = (X_1, X_2, \ldots, X_N) \in \{0,1\}^N$
  - Bob has index $i \in [N]$, and wants to find X_i
  - Lower bound of $\Omega(N)$ bits
- Set $k = \sqrt{N}$
- Fix a bijection $[k] \times [k] \to [N]$
- Introduce $2k$ vertices
  - $v_1, v_2, \ldots, v_k$
  - $w_1, w_2, \ldots, w_k$
- For each $(i,j) \in [k] \times [k]$
  - Alice adds an edge $v_i - w_j$ iff $X_{i,j} = 1$
- Let Bob's index be (i^*, j^*)
- For each $(i,j) \in [k] \times [k]$ such that $i \neq i^*$ and $j \neq j^*$
  - Bob adds two leaves each to v_i and w_j



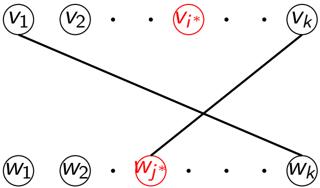
$\mathrm{VC}(G) = 2k - 2$ if and only if $X_{i^*,j^*} = 0$

Parameterized Streaming Algorithms

$\Omega(k^2)$ lower bound for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh [2015]

- INDEX problem
  - Alice has $X = (X_1, X_2, \ldots, X_N) \in \{0,1\}^N$
  - Bob has index $i \in [N]$, and wants to find X_i
  - Lower bound of $\Omega(N)$ bits
- Set $k = \sqrt{N}$
- Fix a bijection $[k] \times [k] \to [N]$
- Introduce $2k$ vertices
  - $v_1, v_2, \ldots, v_k$
  - $w_1, w_2, \ldots, w_k$
- For each $(i,j) \in [k] \times [k]$
  - Alice adds an edge $v_i - w_j$ iff $X_{i,j} = 1$
- Let Bob's index be (i^*, j^*)
- For each $(i,j) \in [k] \times [k]$ such that $i \neq i^*$ and $j \neq j^*$
  - Bob adds two leaves each to v_i and w_j



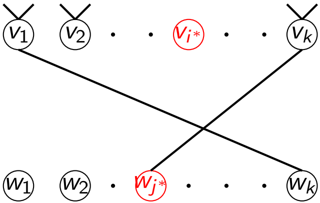
VC$(G) = 2k - 2$ if and only if $X_{i^*,j^*} = 0$

Parameterized Streaming Algorithms

$\Omega(k^2)$ lower bound for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh [2015]

- INDEX problem
  - Alice has $X = (X_1, X_2, \ldots, X_N) \in \{0, 1\}^N$
  - Bob has index $i \in [N]$, and wants to find X_i
  - Lower bound of $\Omega(N)$ bits
- Set $k = \sqrt{N}$
- Fix a bijection $[k] \times [k] \to [N]$
- Introduce $2k$ vertices
  - $v_1, v_2, \ldots, v_k$
  - $w_1, w_2, \ldots, w_k$
- For each $(i, j) \in [k] \times [k]$
  - Alice adds an edge $v_i - w_j$ iff $X_{i,j} = 1$
- Let Bob's index be (i^*, j^*)
- For each $(i, j) \in [k] \times [k]$ such that $i \neq i^*$ and $j \neq j^*$
  - Bob adds two leaves each to v_i and w_j



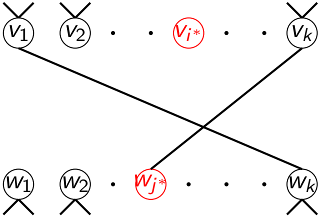
VC$(G) = 2k - 2$ if and only if $X_{i^*,j^*} = 0$

Parameterized Streaming Algorithms

$\Omega(k^2)$ lower bound for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh [2015]

- INDEX problem
  - Alice has $X = (X_1, X_2, \ldots, X_N) \in \{0,1\}^N$
  - Bob has index $i \in [N]$, and wants to find X_i
  - Lower bound of $\Omega(N)$ bits
- Set $k = \sqrt{N}$
- Fix a bijection $[k] \times [k] \to [N]$
- Introduce $2k$ vertices
  - $v_1, v_2, \ldots, v_k$
  - $w_1, w_2, \ldots, w_k$
- For each $(i,j) \in [k] \times [k]$
  - Alice adds an edge $v_i - w_j$ iff $X_{i,j} = 1$
- Let Bob's index be (i^*, j^*)
- For each $(i,j) \in [k] \times [k]$ such that $i \neq i^*$ and $j \neq j^*$
  - Bob adds two leaves each to v_i and w_j



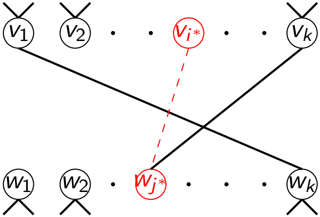
$\mathrm{VC}(G) = 2k - 2$ if and only if $X_{i^*, j^*} = 0$

Parameterized Streaming Algorithms

$\Omega(k^2)$ lower bound for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh [2015]

- INDEX problem
  - Alice has $X = (X_1, X_2, \ldots, X_N) \in \{0,1\}^N$
  - Bob has index $i \in [N]$, and wants to find X_i
  - Lower bound of $\Omega(N)$ bits
- Set $k = \sqrt{N}$
- Fix a bijection $[k] \times [k] \to [N]$
- Introduce $2k$ vertices
  - $v_1, v_2, \ldots, v_k$
  - $w_1, w_2, \ldots, w_k$
- For each $(i,j) \in [k] \times [k]$
  - Alice adds an edge $v_i - w_j$ iff $X_{i,j} = 1$
- Let Bob's index be (i^*, j^*)
- For each $(i,j) \in [k] \times [k]$ such that $i \neq i^*$ and $j \neq j^*$
  - Bob adds two leaves each to v_i and w_j



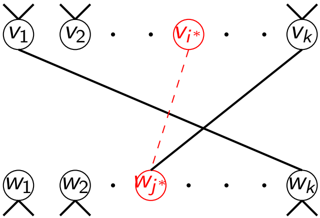
$\text{VC}(G) = 2k - 2$ if and only if $X_{i^*,j^*} = 0$

Parameterized Streaming Algorithms

$\Omega(k^2)$ lower bound for k-VC in insertion-only streams

C., Cormode, Hajiaghayi, Monemizadeh [2015]

- INDEX problem
  - Alice has $X = (X_1, X_2, \dots, X_N) \in \{0,1\}^N$
  - Bob has index $i \in [N]$, and wants to find X_i
  - Lower bound of $\Omega(N)$ bits
- Set $k = \sqrt{N}$
- Fix a bijection $[k] \times [k] \to [N]$
- Introduce $2k$ vertices
  - $v_1, v_2, \dots, v_k$
  - $w_1, w_2, \dots, w_k$
- For each $(i,j) \in [k] \times [k]$
  - Alice adds an edge $v_i - w_j$ iff $X_{i,j} = 1$
- Let Bob's index be (i^*, j^*)
- For each $(i,j) \in [k] \times [k]$ such that $i \neq i^*$ and $j \neq j^*$
  - Bob adds two leaves each to v_i and w_j



$\text{VC}(G) = 2k - 2$ if and only if $X_{i^*,j^*} = 0$

Parameterized Streaming algorithms

Can we handle edge-deletions?

Parameterized Streaming algorithms

Can we handle edge-deletions?

- C., Cormode, Hajiaghayi, Monemizadeh ['15]
  - $O(nk \cdot \log^{O(1)} n)$ space algorithm for k-VC in insertion-deletion streams

Parameterized Streaming algorithms

Can we handle edge-deletions?

- C., Cormode, Hajiaghayi, Monemizadeh ['15]
  - $O(nk \cdot \log^{O(1)} n)$ space algorithm for k-VC in insertion-deletion streams
- C., Esfandiari, Cormode, Hajiaghayi, Monemizadeh ['15]
  - Promise: VC is $\leq k$ at every timestamp

Parameterized Streaming algorithms

Can we handle edge-deletions?

- C., Cormode, Hajiaghayi, Monemizadeh ['15]
  - $O(nk \cdot \log^{O(1)} n)$ space algorithm for k-VC in insertion-deletion streams
- C., Esfandiari, Cormode, Hajiaghayi, Monemizadeh ['15]
  - Promise: VC is $\leq k$ at every timestamp
  - $O(k^2 \cdot \log^{O(1)} n)$ space algorithm for k-VC in insertion-deletion streams

Parameterized Streaming algorithms

Can we handle edge-deletions?

- C., Cormode, Hajiaghayi, Monemizadeh ['15]
  - $O(nk \cdot \log^{O(1)} n)$ space algorithm for k-VC in insertion-deletion streams
- C., Esfandiari, Cormode, Hajiaghayi, Monemizadeh ['15]
  - Promise: VC is $\leq k$ at every timestamp
  - $O(k^2 \cdot \log^{O(1)} n)$ space algorithm for k-VC in insertion-deletion streams
- C., Cormode, Esfandiari, Hajiaghayi, Mcgregor, Monemizadeh, Vorotnikova ['16]
  - Promise: VC is $\leq k$ at end of stream

Parameterized Streaming algorithms

Can we handle edge-deletions?

- C., Cormode, Hajiaghayi, Monemizadeh ['15]
  - $O(nk \cdot \log^{O(1)} n)$ space algorithm for k-VC in insertion-deletion streams
- C., Esfandiari, Cormode, Hajiaghayi, Monemizadeh ['15]
  - Promise: VC is $\leq k$ at every timestamp
  - $O(k^2 \cdot \log^{O(1)} n)$ space algorithm for k-VC in insertion-deletion streams
- C., Cormode, Esfandiari, Hajiaghayi, Mcgregor, Monemizadeh, Vorotnikova ['16]
  - Promise: VC is $\leq k$ at end of stream
  - $O(k^2 \cdot \log^{O(1)} n)$ space for k-VC in insertion-deletion streams

Parameterized Streaming algorithms

Can we handle edge-deletions?

- C., Cormode, Hajiaghayi, Monemizadeh ['15]
  - $O(nk \cdot \log^{O(1)} n)$ space algorithm for k-VC in insertion-deletion streams
- C., Esfandiari, Cormode, Hajiaghayi, Monemizadeh ['15]
  - Promise: VC is $\leq k$ at every timestamp
  - $O(k^2 \cdot \log^{O(1)} n)$ space algorithm for k-VC in insertion-deletion streams
- C., Cormode, Esfandiari, Hajiaghayi, Mcgregor, Monemizadeh, Vorotnikova ['16]
  - Promise: VC is $\leq k$ at end of stream
  - $O(k^2 \cdot \log^{O(1)} n)$ space for k-VC in insertion-deletion streams
  - Also works for Maximum Matching

Parameterized Streaming algorithms

Can we handle edge-deletions?

- C., Cormode, Hajiaghayi, Monemizadeh ['15]
  - $O(nk \cdot \log^{O(1)} n)$ space algorithm for k-VC in insertion-deletion streams
- C., Esfandiari, Cormode, Hajiaghayi, Monemizadeh ['15]
  - Promise: VC is $\leq k$ at every timestamp
  - $O(k^2 \cdot \log^{O(1)} n)$ space algorithm for k-VC in insertion-deletion streams
- C., Cormode, Esfandiari, Hajiaghayi, Mcgregor, Monemizadeh, Vorotnikova ['16]
  - Promise: VC is $\leq k$ at end of stream
  - $O(k^2 \cdot \log^{O(1)} n)$ space for k-VC in insertion-deletion streams
  - Also works for Maximum Matching
  - Generalizes to d-uniform hypergraphs

Parameterized Streaming algorithms

Can we handle edge-deletions?

- C., Cormode, Hajiaghayi, Monemizadeh ['15]
  - $O(nk \cdot \log^{O(1)} n)$ space algorithm for k-VC in insertion-deletion streams
- C., Esfandiari, Cormode, Hajiaghayi, Monemizadeh ['15]
  - Promise: VC is $\leq k$ at every timestamp
  - $O(k^2 \cdot \log^{O(1)} n)$ space algorithm for k-VC in insertion-deletion streams
- C., Cormode, Esfandiari, Hajiaghayi, Mcgregor, Monemizadeh, Vorotnikova ['16]
  - Promise: VC is $\leq k$ at end of stream
  - $O(k^2 \cdot \log^{O(1)} n)$ space for k-VC in insertion-deletion streams
  - Also works for Maximum Matching
  - Generalizes to d-uniform hypergraphs
  - Sketch on next slide ...

Parameterized Streaming algorithms

$O(k^2 \cdot \text{poly} \log n)$ space algorithm for k-VC in insertion-deletion streams

C., Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova ['16]

- Promise: VC $\leq k$ at end of stream

Parameterized Streaming algorithms

$O(k^2 \cdot \text{poly} \log n)$ space algorithm for k-VC in insertion-deletion streams

C., Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova ['16]

- Promise: VC $\leq k$ at end of stream
- Color vertices using $O(k)$ colors
  - Pick coloring from a family of pairwise independent hash functions

Parameterized Streaming algorithms

$O(k^2 \cdot \operatorname{poly}\log n)$ space algorithm for k-VC in insertion-deletion streams

C., Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova ['16]

- Promise: VC $\leq k$ at end of stream
- Color vertices using $O(k)$ colors
  - Pick coloring from a family of pairwise independent hash functions
- Forget edges within a color class

Parameterized Streaming algorithms

$O(k^2 \cdot \text{poly}\log n)$ space algorithm for k-VC in insertion-deletion streams

C., Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova ['16]

- Promise: VC $\leq k$ at end of stream
- Color vertices using $O(k)$ colors
  - Pick coloring from a family of pairwise independent hash functions
- Forget edges within a color class
- For every pair of color classes
  - Pick one edge u.a.r using ℓ_0-sampler

Parameterized Streaming algorithms

$O(k^2 \cdot \operatorname{poly}\log n)$ space algorithm for k-VC in insertion-deletion streams

C., Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova ['16]

- Promise: VC $\leq k$ at end of stream
- Color vertices using $O(k)$ colors
  - Pick coloring from a family of pairwise independent hash functions
- Forget edges within a color class
- For every pair of color classes
  - Pick one edge u.a.r using ℓ_0-sampler
- Let G' be resulting graph

Parameterized Streaming algorithms

$O(k^2 \cdot \text{poly}\log n)$ space algorithm for k-VC in insertion-deletion streams

C., Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova ['16]

- Promise: VC $\leq k$ at end of stream
- Color vertices using $O(k)$ colors
  - Pick coloring from a family of pairwise independent hash functions
- Forget edges within a color class
- For every pair of color classes
  - Pick one edge u.a.r using ℓ_0-sampler
- Let G' be resulting graph
- With probability $1/2$ it holds that

Parameterized Streaming algorithms

$O(k^2 \cdot \operatorname{poly}\log n)$ space algorithm for k-VC in insertion-deletion streams

C., Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova ['16]

- Promise: VC $\leq k$ at end of stream
- Color vertices using $O(k)$ colors
  - Pick coloring from a family of pairwise independent hash functions
- Forget edges within a color class
- For every pair of color classes
  - Pick one edge u.a.r using ℓ_0-sampler
- Let G' be resulting graph
- With probability $1/2$ it holds that
  - $MM(G) = MM(G')$
  - $VC(G) = VC(G')$
- G' is randomized kernel

Parameterized Streaming algorithms

$O(k^2 \cdot \text{poly}\log n)$ space algorithm for k-VC in insertion-deletion streams

C., Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova ['16]

- Promise: VC $\leq k$ at end of stream
- Color vertices using $O(k)$ colors
  - Pick coloring from a family of pairwise independent hash functions
- Forget edges within a color class
- For every pair of color classes
  - Pick one edge u.a.r using ℓ_0-sampler
- Let G' be resulting graph
- With probability $1/2$ it holds that
  - $MM(G) = MM(G')$
  - $VC(G) = VC(G')$
- G' is randomized kernel
- Space bound

Parameterized Streaming algorithms

$O(k^2 \cdot \text{poly}\log n)$ space algorithm for k-VC in insertion-deletion streams

C., Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova ['16]

- Promise: VC $\leq k$ at end of stream
- Color vertices using $O(k)$ colors
  - Pick coloring from a family of pairwise independent hash functions
- Forget edges within a color class
- For every pair of color classes
  - Pick one edge u.a.r using ℓ_0-sampler
- Let G' be resulting graph
- With probability $1/2$ it holds that
  - $MM(G) = MM(G')$
  - $VC(G) = VC(G')$
- G' is randomized kernel
- Space bound
  - There are $O(k^2)$ pairs of color classes

Parameterized Streaming algorithms

$O(k^2 \cdot \text{poly}\log n)$ space algorithm for k-VC in insertion-deletion streams

C., Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova ['16]

- Promise: VC $\leq k$ at end of stream
- Color vertices using $O(k)$ colors
  - Pick coloring from a family of pairwise independent hash functions
- Forget edges within a color class
- For every pair of color classes
  - Pick one edge u.a.r using ℓ_0-sampler
- Let G' be resulting graph
- With probability $1/2$ it holds that
  - $MM(G) = MM(G')$
  - $VC(G) = VC(G')$
- G' is randomized kernel
- Space bound
  - There are $O(k^2)$ pairs of color classes
  - For each pair of color classes we use a ℓ_0-sampler

Parameterized Streaming algorithms

$O(k^2 \cdot \text{poly}\log n)$ space algorithm for k-MM in insertion-deletion streams

C., Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova ['16]

- Two applications in non-parameterized streaming algorithms which use this algorithm as sub-routine

Parameterized Streaming algorithms

$O(k^2 \cdot \text{poly}\log n)$ space algorithm for k-MM in insertion-deletion streams

C., Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova ['16]

- Two applications in non-parameterized streaming algorithms which use this algorithm as sub-routine

- $O(n^{1/3})$**-approximation for MM in dynamic streams in** $O(n \cdot \textbf{poly}\log n)$ **space**

Parameterized Streaming algorithms

$O(k^2 \cdot \text{poly} \log n)$ space algorithm for k-MM in insertion-deletion streams

C., Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova ['16]

- Two applications in non-parameterized streaming algorithms which use this algorithm as sub-routine
- $O(n^{1/3})$**-approximation for MM in dynamic streams in** $O(n \cdot \textbf{poly} \log n)$ **space**
  - First sublinear approximation for dynamic streams in semi-streaming model

Parameterized Streaming algorithms

$O(k^2 \cdot \text{poly}\log n)$ space algorithm for k-MM in insertion-deletion streams

C., Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova ['16]

- Two applications in non-parameterized streaming algorithms which use this algorithm as sub-routine
- $O(n^{1/3})$**-approximation for MM in dynamic streams in** $O(n \cdot \textbf{poly}\log n)$ **space**
  - First sublinear approximation for dynamic streams in semi-streaming model
- $O(1)$**-approximation for estimating MM size in planar dynamic streams in** $O(n^{4/5})$ **space**

Parameterized Streaming algorithms

$O(k^2 \cdot \text{poly}\log n)$ space algorithm for k-MM in insertion-deletion streams

C., Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova ['16]

- Two applications in non-parameterized streaming algorithms which use this algorithm as sub-routine
- **$O(n^{1/3})$-approximation for MM in dynamic streams in $O(n \cdot \textbf{poly}\log n)$ space**
  - First sublinear approximation for dynamic streams in semi-streaming model
- **$O(1)$-approximation for estimating MM size in planar dynamic streams in $O(n^{4/5})$ space**
  - First sublinear space constant-factor approximation for estimating MM size in planar dynamic streams

Parameterized Streaming algorithms

$O(k^2 \cdot \text{poly}\log n)$ space algorithm for k-MM in insertion-deletion streams

C., Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova ['16]

Implemented on some real-world BIG data...

Parameterized Streaming algorithms

$O(k^2 \cdot \text{poly}\log n)$ space algorithm for k-MM in insertion-deletion streams

C., Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova ['16]

Implemented on some real-world BIG data...

BigDND: Big Dynamic Network Data

[Erik Demaine](#) (MIT) & [MohammadTaghi Hajiaghayi](#) (UMD)

Networks are everywhere, and there is an increasing amount of data about networks viewed as graphs: nodes and edges/connections. But this data typically ignores a third key component of networks: time. This repository provides **free, big datasets for real-world networks** viewed as a dynamic (multi)graph, with two types of temporal data:

1. A timeseries of **instantaneous edge events**, such as messages sent between people. Many such events can occur between the same pair of nodes.
2. Timestamped **edge insertions and edge deletions**, such as friending and defriending in a social network. Generally only one such edge can exist at any specific time, but the same edge can be added and deleted multiple times.

Our hope is that these datasets will promote new research into the dynamics of complex networks, improving our understanding of their behavior, and helping the community to experimentally evaluate their big-data algorithms: approximation, fixed-parameter, external-memory, streaming, and network-analysis algorithms.

Help us:

- If you have a **dynamic network dataset**, email us at *dnd (at) csail.mit.edu* with a brief description about the data, its format, its license, and how/where to download it. We will link to it with appropriate credit/citation.
- If you have interesting **visualizations and/or analysis** of these data sets, email us at *dnd (at) csail.mit.edu* and we will post it with appropriate credit/citation.

http://projects.csail.mit.edu/dnd/

Parameterized Streaming algorithms

Other examples

Parameterized Streaming algorithms

Other examples

- Some problems have $\Omega(n)$ lower bound for constant k

Parameterized Streaming algorithms

Other examples

- Some problems have $\Omega(n)$ lower bound for constant k
  - Rules out $f(k) \cdot n^{1-\beta}$ space algorithms for any $\beta > 0$

Parameterized Streaming algorithms

Other examples

- Some problems have $\Omega(n)$ lower bound for constant k
  - Rules out $f(k) \cdot n^{1-\beta}$ space algorithms for any $\beta > 0$
- k-FVS (Feedback Vertex Set)

Parameterized Streaming algorithms

Other examples

- Some problems have $\Omega(n)$ lower bound for constant k
  - Rules out $f(k) \cdot n^{1-\beta}$ space algorithms for any $\beta > 0$

- k-FVS (Feedback Vertex Set)
  - Is there a set of $\leq k$ vertices whose deletion makes the graph acyclic?

Parameterized Streaming algorithms

Other examples

- Some problems have $\Omega(n)$ lower bound for constant k
  - Rules out $f(k) \cdot n^{1-\beta}$ space algorithms for any $\beta > 0$

- k-FVS (Feedback Vertex Set)
  - Is there a set of $\leq k$ vertices whose deletion makes the graph acyclic?
  - $\Omega(n)$ lower bound for $k = 0$

Parameterized Streaming algorithms

Other examples

- Some problems have $\Omega(n)$ lower bound for constant k
  - Rules out $f(k) \cdot n^{1-\beta}$ space algorithms for any $\beta > 0$

- k-FVS (Feedback Vertex Set)
  - Is there a set of $\leq k$ vertices whose deletion makes the graph acyclic?
  - $\Omega(n)$ lower bound for $k = 0$
  - Observation: FVS $\leq k$ implies graph can have $\leq O(k \cdot n)$ edges

Parameterized Streaming algorithms

Other examples

- Some problems have $\Omega(n)$ lower bound for constant k
  - Rules out $f(k) \cdot n^{1-\beta}$ space algorithms for any $\beta > 0$

- k-FVS (Feedback Vertex Set)
  - Is there a set of $\leq k$ vertices whose deletion makes the graph acyclic?
  - $\Omega(n)$ lower bound for $k = 0$
  - Observation: FVS $\leq k$ implies graph can have $\leq O(k \cdot n)$ edges

- k-Path

Parameterized Streaming algorithms

Other examples

- Some problems have $\Omega(n)$ lower bound for constant k
  - Rules out $f(k) \cdot n^{1-\beta}$ space algorithms for any $\beta > 0$

- k-FVS (Feedback Vertex Set)
  - Is there a set of $\leq k$ vertices whose deletion makes the graph acyclic?
  - $\Omega(n)$ lower bound for $k = 0$
  - Observation: FVS $\leq k$ implies graph can have $\leq O(k \cdot n)$ edges

- k-Path
  - Is there a path of length $\geq k$

Parameterized Streaming algorithms

Other examples

- Some problems have $\Omega(n)$ lower bound for constant k
  - Rules out $f(k) \cdot n^{1-\beta}$ space algorithms for any $\beta > 0$

- k-FVS (Feedback Vertex Set)
  - Is there a set of $\leq k$ vertices whose deletion makes the graph acyclic?
  - $\Omega(n)$ lower bound for $k = 0$
  - Observation: FVS $\leq k$ implies graph can have $\leq O(k \cdot n)$ edges

- k-Path
  - Is there a path of length $\geq k$
  - $\Omega(n)$ lower bound for $k = 3$

Parameterized Streaming algorithms

Other examples

- Some problems have $\Omega(n)$ lower bound for constant k
  - Rules out $f(k) \cdot n^{1-\beta}$ space algorithms for any $\beta > 0$

- k-FVS (Feedback Vertex Set)
  - Is there a set of $\leq k$ vertices whose deletion makes the graph acyclic?
  - $\Omega(n)$ lower bound for $k = 0$
  - Observation: FVS $\leq k$ implies graph can have $\leq O(k \cdot n)$ edges

- k-Path
  - Is there a path of length $\geq k$
  - $\Omega(n)$ lower bound for $k = 3$
  - Observation: At least nk edges implies existence of k-path

Looking forward ....

Looking forward ....

- I'm not aware of that many results on parameterized streaming

Looking forward ....

- I'm not aware of that many results on parameterized streaming
- Parameter does not have to be size of solution!

Looking forward ....

- I'm not aware of that many results on parameterized streaming

- Parameter does not have to be size of solution!
  - Treewidth
  - Max degree
  - Girth

Looking forward ....

- I'm not aware of that many results on parameterized streaming

- Parameter does not have to be size of solution!
  - Treewidth
  - Max degree
  - Girth
  - ......

Looking forward ....

- I'm not aware of that many results on parameterized streaming
- Parameter does not have to be size of solution!
  - Treewidth
  - Max degree
  - Girth
  - ......
- Lower bounds $\Rightarrow$ birth of new (types of) algorithms

Looking forward ....

- I'm not aware of that many results on parameterized streaming
- Parameter does not have to be size of solution!
  - Treewidth
  - Max degree
  - Girth
  - ......
- Lower bounds $\Rightarrow$ birth of new (types of) algorithms
- Let X be a graph problem with an $\Omega(n)$ lower bound

Looking forward ....

- I'm not aware of that many results on parameterized streaming
- Parameter does not have to be size of solution!
  - Treewidth
  - Max degree
  - Girth
  - ......
- Lower bounds $\Rightarrow$ birth of new (types of) algorithms
- Let X be a graph problem with an $\Omega(n)$ lower bound
  - Say can design $f(k) \cdot \log^{O(1)} n$ space algorithm for some parameter k

Looking forward ....

- I'm not aware of that many results on parameterized streaming
- Parameter does not have to be size of solution!
  - Treewidth
  - Max degree
  - Girth
  - ......
- Lower bounds $\Rightarrow$ birth of new (types of) algorithms
- Let X be a graph problem with an $\Omega(n)$ lower bound
  - Say can design $f(k) \cdot \log^{O(1)} n$ space algorithm for some parameter k
  - This means that the parameter k was a barrier to small-space algorithms

Looking forward ....

- I'm not aware of that many results on parameterized streaming
- Parameter does not have to be size of solution!
  - Treewidth
  - Max degree
  - Girth
  - ......
- Lower bounds $\Rightarrow$ birth of new (types of) algorithms
- Let X be a graph problem with an $\Omega(n)$ lower bound
  - Say can design $f(k) \cdot \log^{O(1)} n$ space algorithm for some parameter k
  - This means that the parameter k was a barrier to small-space algorithms
  - Helps to pinpoint the reason(s) for intractability!

Looking forward ....

- I'm not aware of that many results on parameterized streaming
- Parameter does not have to be size of solution!
  - Treewidth
  - Max degree
  - Girth
  - ......
- Lower bounds $\Rightarrow$ birth of new (types of) algorithms
- Let X be a graph problem with an $\Omega(n)$ lower bound
  - Say can design $f(k) \cdot \log^{O(1)} n$ space algorithm for some parameter k
  - This means that the parameter k was a barrier to small-space algorithms
  - Helps to pinpoint the reason(s) for intractability!
- Choose your favorite (graph) problems and parameters!

Thank You

Questions?