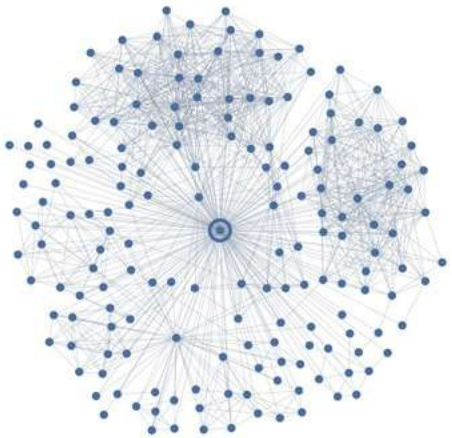
Towards a Theory of Parameterized Streaming Algorithms

Graham Cormode
Rajesh Chitnis

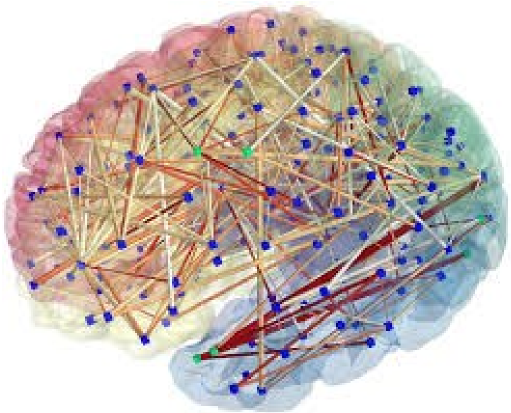
THE UNIVERSITY OF WARWICK

Parameterized Streaming Algorithms

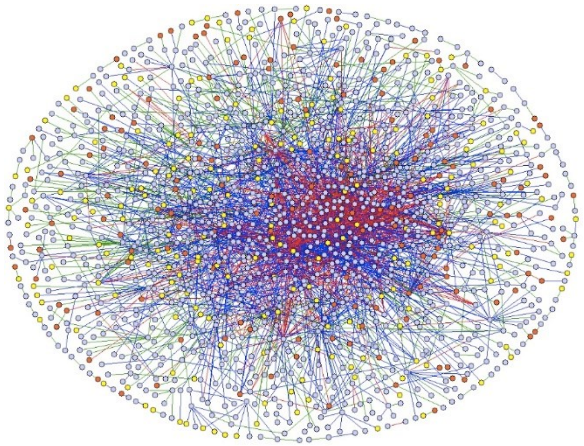
We increasingly have to deal with huge graphs…



Facebook graph
- 10^9 nodes



Brain graph
- 10^9 nodes



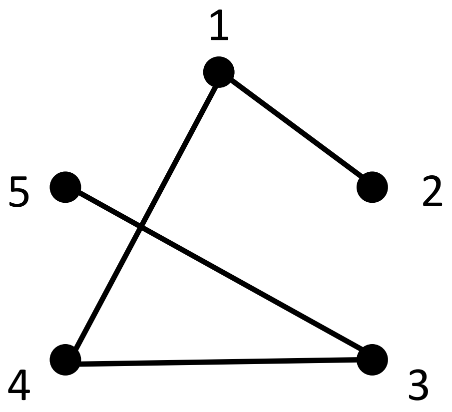
Web Graph
- 2^{32} nodes

Google Maps in USA
- 10^8 intersection nodes

- It is inconvenient or impossible to store the whole input for random access
- "Solved" problems become hard under different models of data access
  - E.g. External memory, MapReduce, Streaming…

Parameterized Streaming Algorithms

- The paradigm of streaming algorithms is one attempt to deal with Big Data
- The streaming model (for graphs) is as follows:
  - The vertex set $V = \{1,2,\dots,n\}$ is fixed, and known in advance
  - The edges arrive one-by-one (in arbitrary order)
  - For each edge arrival, we need to make a (fast) decision what information to store
  - Cannot (do not want to) store all the edges



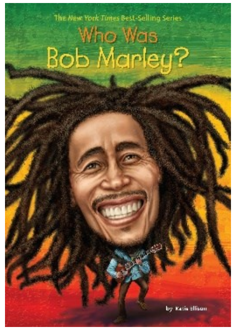
- We allow unbounded computation at end of the stream
- Which graph problems can we solve efficiently in this model?
  - Naïve algorithm for *any* graph problem uses $O(n^2)$ bits by storing whole adjacency matrix

Parameterized Streaming Algorithms

- Recall that the naïve algorithm for any graph problem uses $O(n^2)$ bits
- **Bad News** : Many graph problems have a lower bound of $\Omega(n^2)$ space in streaming model
  - E.g. Does the given graph have any triangle?
- Typically use communication complexity to show lower bounds for streaming algorithms
- INDEX problem: Alice has string $X \in \{0,1\}^N$, Bob has index $i \in [N]$, want to find ith bit of X
  - Lower bound of $\Omega(N)$ if Alice can send only one message to Bob, even with randomization
- Communication complexity reductions: show that a streaming algorithm would solve INDEX

10010110

One-way communication from Alice to Bob

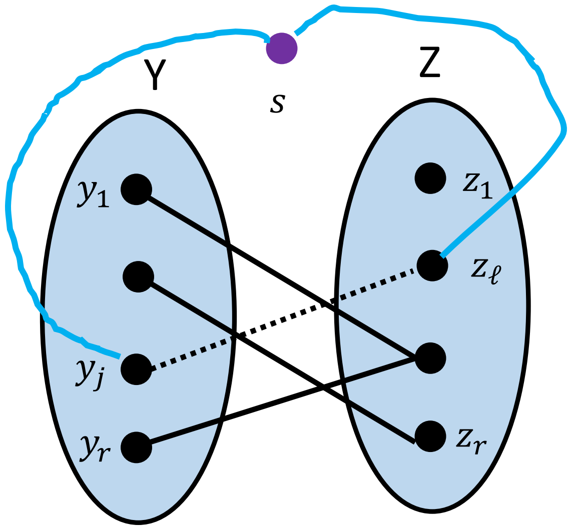


$i = 5$

Parameterized Streaming Algorithms

- Sketch of a simple INDEX reduction for triangle detection:
- Alice adds edges between Y and Z according to her string X
  - Then she sends her data structure to Bob
- Bob has an index $I \in N$ corresponding to some $(j, \ell) \in [r] \times [r]$
  - Bob adds a new vertex s and the edges (s, y_j) and (s, z_ℓ)

Let $N = r^2$



The resulting graph has a triangle iff the edge (y_j, z_ℓ) is present, i.e., I^{th} bit of X is 1

Parameterized Streaming Algorithms

- **Bad News** : Many graph problems require $\Omega(n^2)$ space in streaming model
- How can we cope with this (space) intractability?

Fine-grained understanding
via parameterized analysis

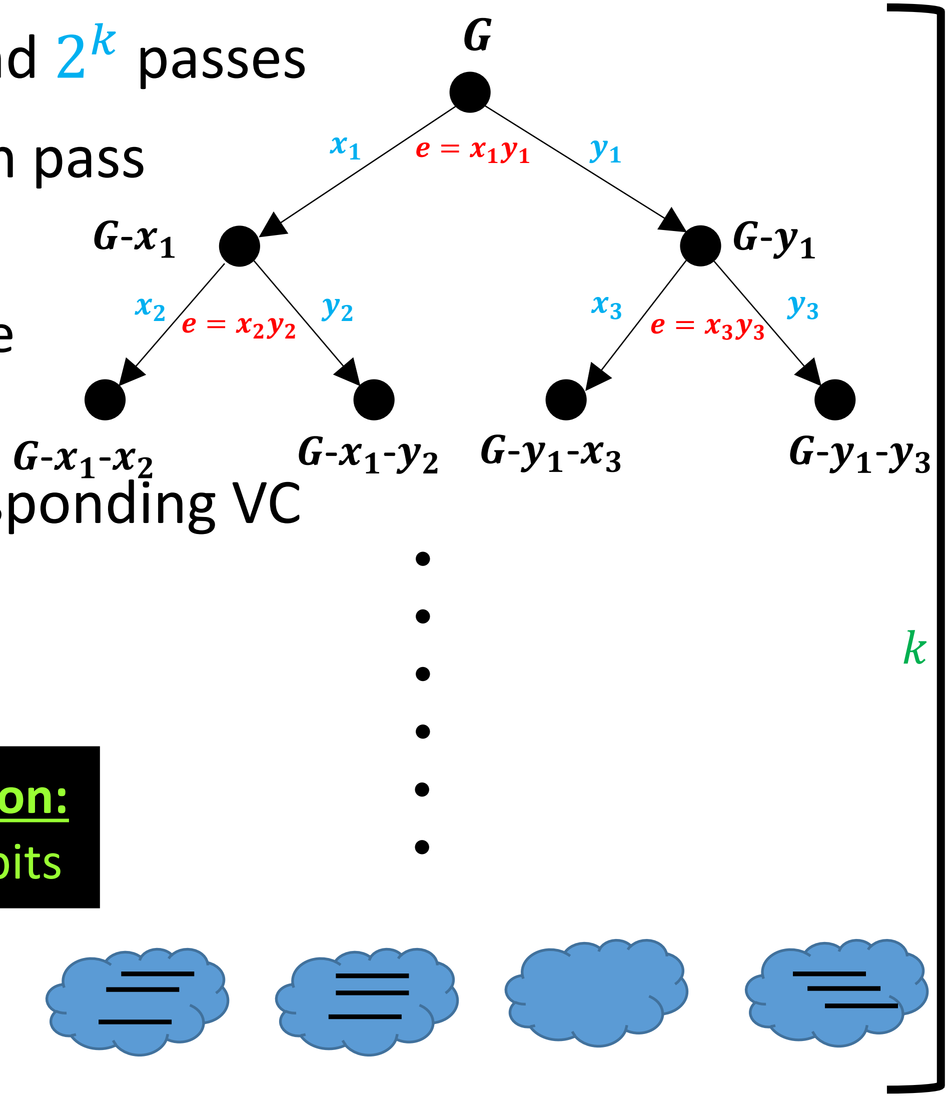
BIG
Time

BIG
Data

- **<u>Feigenbaum et al. [ICALP '04]</u>**: Finding (size of) a min VC needs $\Omega(n^2)$ space
- But how much space does k-VC need?
  - We design a streaming algorithm in $O(k \cdot \log n)$ bits (with 2^k passes over the input)
  - Essentially, the standard branching FPT algorithm in streaming model…

Parameterized Streaming Algorithms

- Streaming algorithm for k-VC with $O(k \cdot \log n)$ bits and 2^k passes
  - Consider all 2^k binary strings from $\{0,1\}^k$, one in each pass
  - The binary search tree has 2^k leaves
    - Each pass corresponds to a root → leaf path in the tree
    - 0 for left branch, and 1 for right branch
  - Algorithm only stores current binary string and corresponding VC
    - Storage is $O(k \cdot \log n)$ bits
    - Optimal if you also want to output a VC!

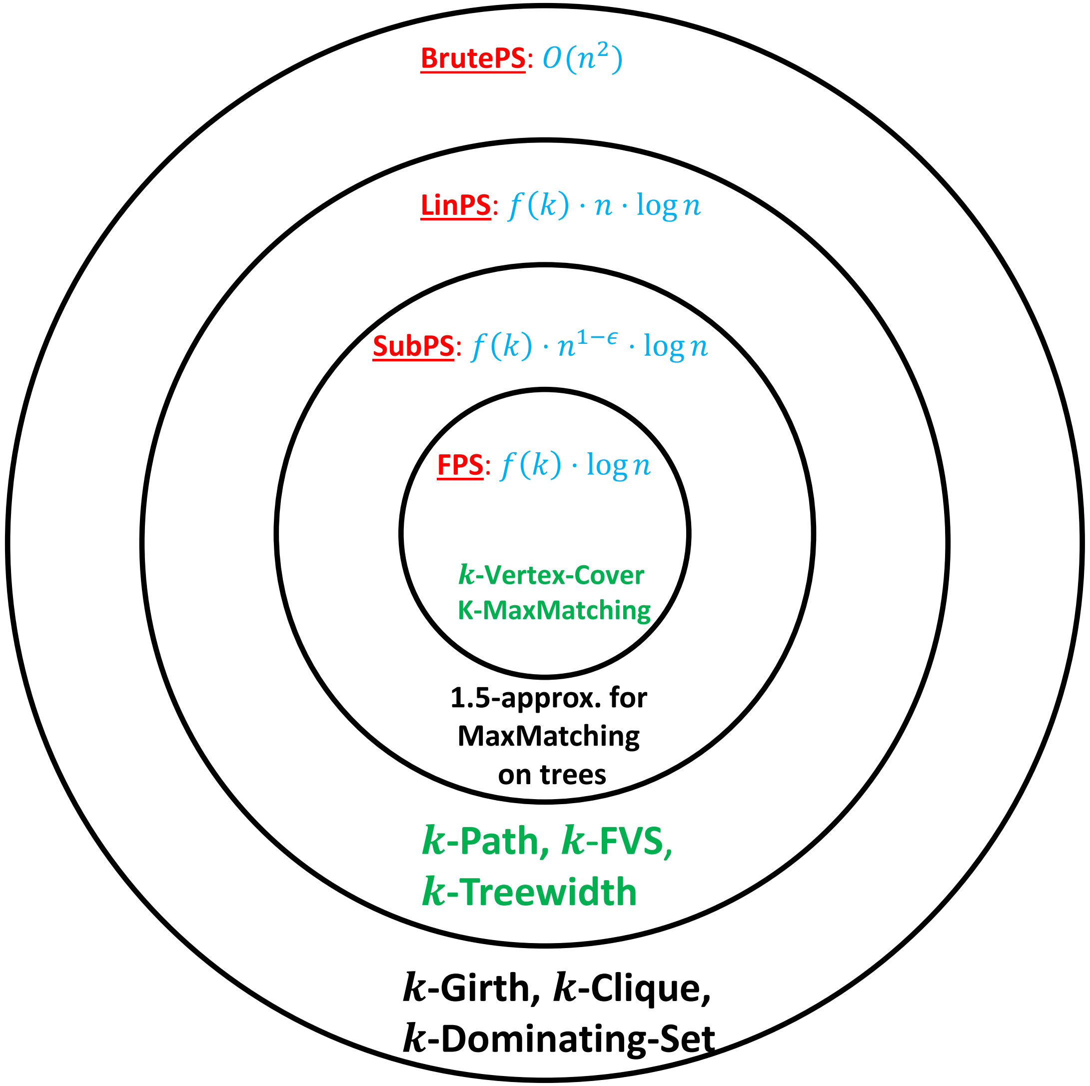


Streaming implementation of FPT algorithm via iterative compression:
$(k \cdot 2^k)$-pass streaming algorithm for k-VC which uses $O(k \cdot \log n)$ bits

Reducing the number of passes: Chitnis et al. [SODA '15] designed a 1-pass streaming algorithm for k-VC using $O(k^2 \cdot \log n)$ bits

Parameterized Streaming Algorithms

Towards a general theory of (space) parameterized streaming algorithms…..



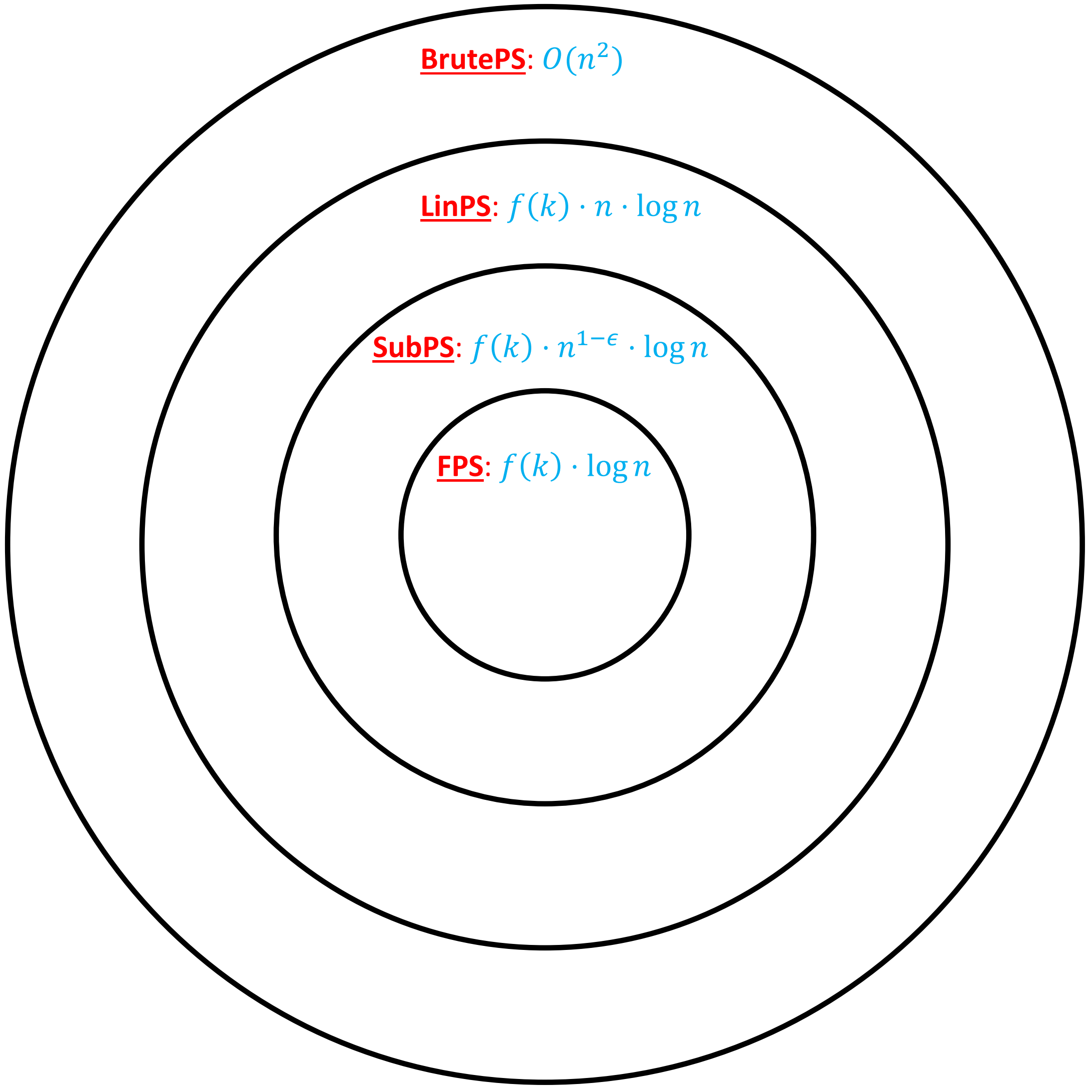
- FPS: Fixed-Parameter Streaming
- SubPS: Sublinear dependence on input n
- LinPS: Linear dependence on input n
- BrutePS: Naïvely storing the whole graph

Goal: Develop algorithms and lower bounds to categorize graph problems in this hierarchy

We study all problems, not just NP-hard ones!

Parameterized Streaming Algorithms

Towards a general theory of (space) parameterized streaming algorithms…..



- FPS: Fixed-Parameter Streaming Algorithms
- SubPS: Sublinear dependence on input n
- LinPS: Linear dependence on input n
- BrutePS: Naïvely storing the whole graph

Picture is a bit more complicated:
Any entry in this landscape is really a 6-tuple

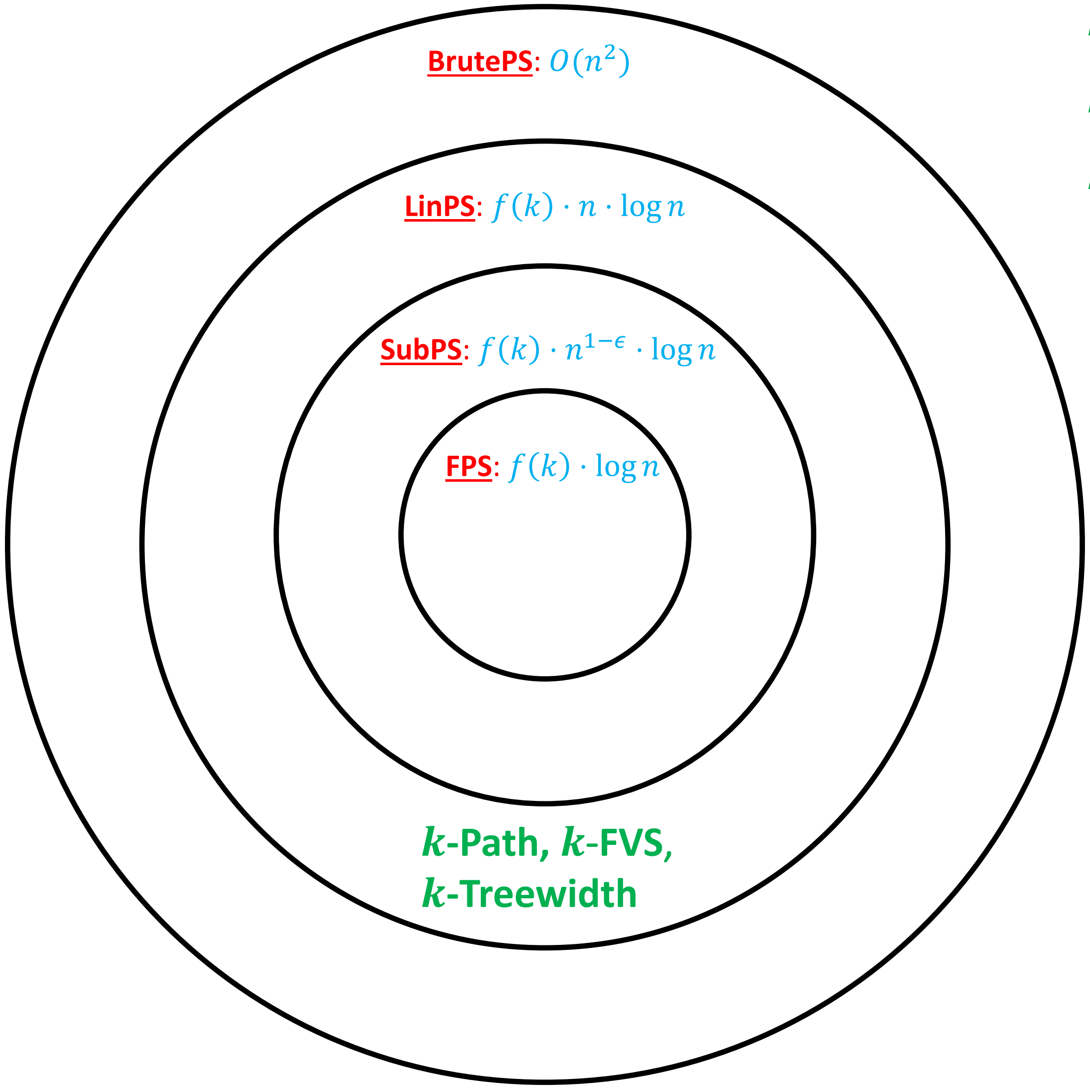
[Problem, Parameter, Approximation Ratio, Type of Stream, Type of Algorithm, # of passes]

Type of Algorithm → Deterministic or Randomized

Type of Stream → Insertion-only or Insertion-deletion

Parameterized Streaming Algorithms

Tight problems for the class LinPS via simple upper bounds



k-Path: If $|E| \geq k \cdot n$ then there is a k-path
k-FVS: If there is a fvs of size k then $|E| \leq k \cdot n$
k-Treewidth: If treewidth is $\leq k$ then $|E| \leq k \cdot n$

Store all edges till we see $(k \cdot n)$ edges
Hence this needs $O(k \cdot n \cdot \log n)$ bits

These problems need $\Omega(n \cdot \log n)$ space (for constant k)
Hence, they are not in SubPS

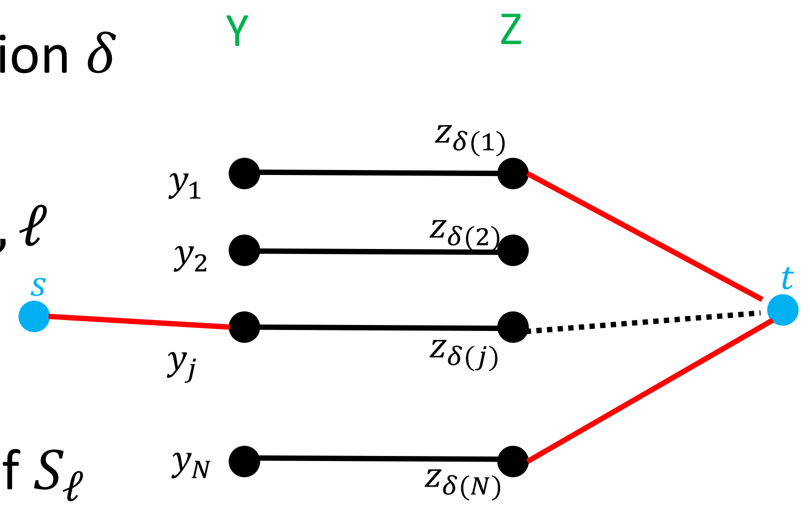
Rules out any algorithm using space $f(k) \cdot o(n \cdot \log n)$ for any function f

Parameterized Streaming Algorithms

$\Omega(n \cdot log\ n)$ bit lower bound for k-Path with $k = 6$

- Hardness reduction: "Small" space streaming algorithm for 6-Path
  $\Rightarrow$ 1- way communication protocol for PERMUTATION of "small" cost
- PERMUTATION problem:
  Alice has a permutation $\delta: [N] \to [N]$ encoded as a bit-string of length $N \cdot \log n$.
  Bob has an index $I \in [N \cdot \log N]$ and wants to find I^{th} bit of δ
  - Sun and Woodruff [APPROX '15]: need $\Omega(N \cdot \log N)$ bits one-way communication

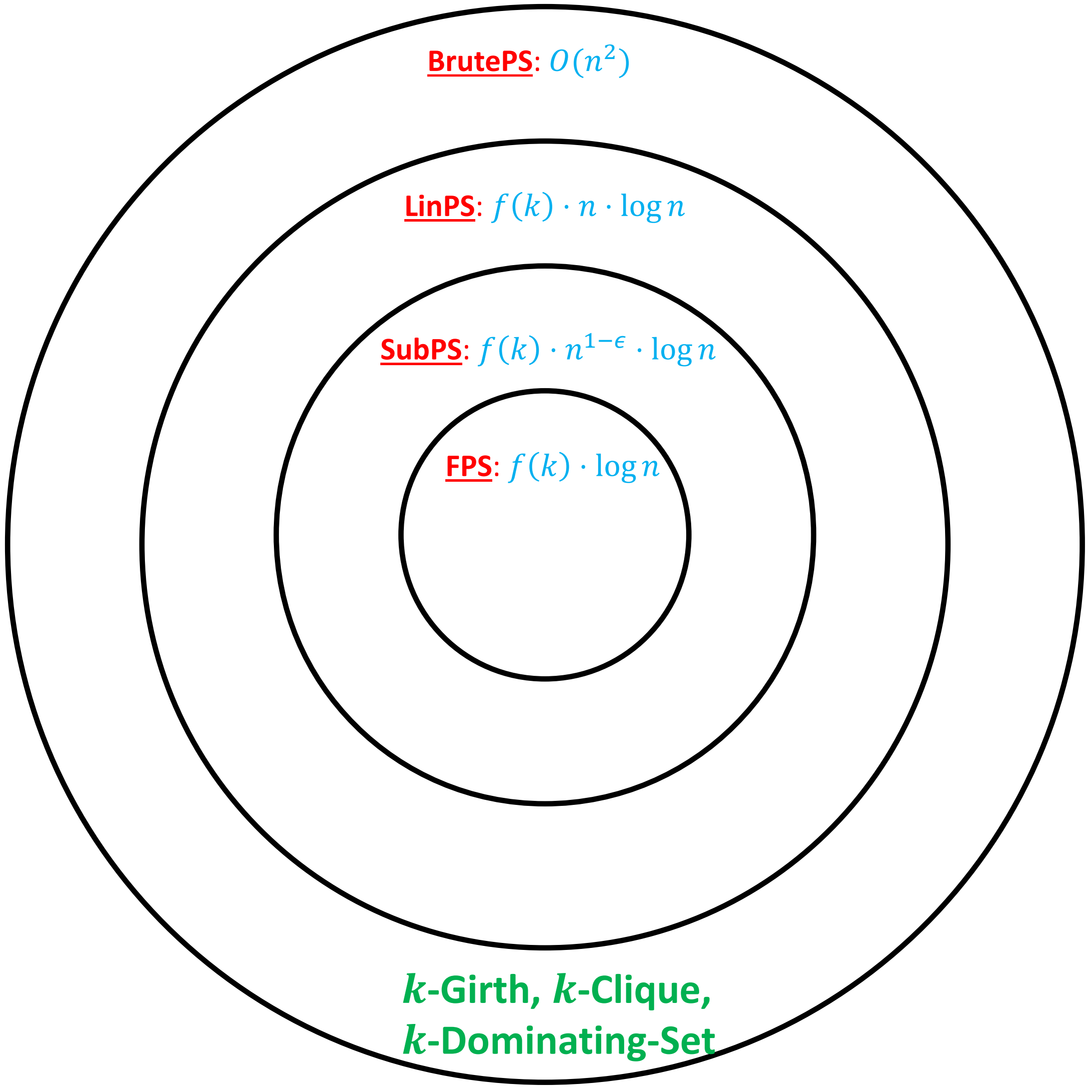
- Alice adds edges between Y and Z according to the permutation δ
  - For each $i \in [N]$ she adds an edge from y_i to $z_{\delta(i)}$
- Bob's index $I \in [N \cdot \log N]$ maps to ℓ^{th}-bit of $\delta(j)$ for some j, ℓ
  - Bob adds a new vertex s, and the edge $s - y_j$
  - Let $S_\ell = \{z_{\delta(r)} : \ell^{th}$-bit of $\delta(r)$ is one $\}$
  - Bob adds new vertex t, and edges from t to each vertex of S_ℓ



The resulting graph has a 6-path iff edge $z_{\delta(j)} \in S_\ell$ is present, i.e., I^{th} bit of X is 1

Parameterized Streaming Algorithms

Tight problems for the class BrutePS



How do we show a problem does not belong to the smaller class LinPS?

- Show $\Omega(n^2)$ bits lower bound for constant k
- Rules out any algorithm using space $f(k) \cdot o(n^2)$
- Next slide gives proof for 3-Girth…

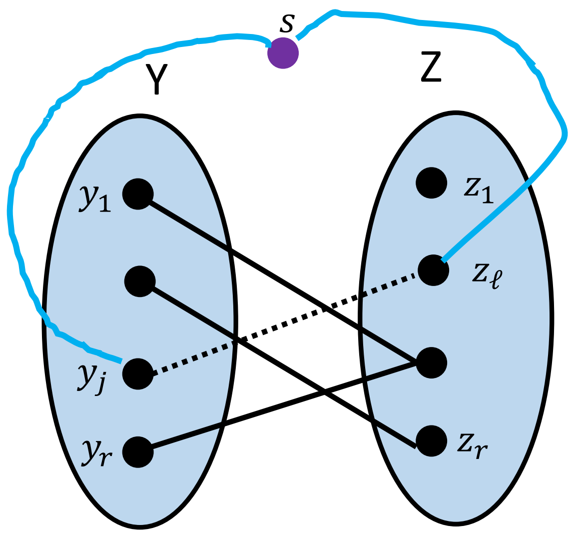
Note that k-Girth is polynomial *time* solvable, but hard in terms of *space*!

Parameterized Streaming Algorithms

$\mathbf{\Omega(n^2)}$ bits lower bound for checking if girth of a graph is $\leq \mathbf{3}$

INDEX problem requires $\Omega(N)$ bits of one-way communication from Alice to Bob
Alice has a string $X \in \{0,1\}^N$.
Bob has an index $I \in [N]$ and wants to find I^{th} bit of X

- Same set up as previously:
  - Let $N = r^2$ and fix a bijection $\phi: [N] \rightarrow [r] \times [r]$
- Alice adds edges between Y and Z according to string X
  - Then she sends her data structure to Bob
- Bob's index $I \in N$ corresponds to some $(j, \ell) \in [r] \times [r]$
  - Bob adds a new vertex s and the edges (s, y_j) and (s, z_ℓ)
- Lower bound of $\Omega(N)$ translates to $\Omega(n^2)$ for 3-girth on graphs with n vertices



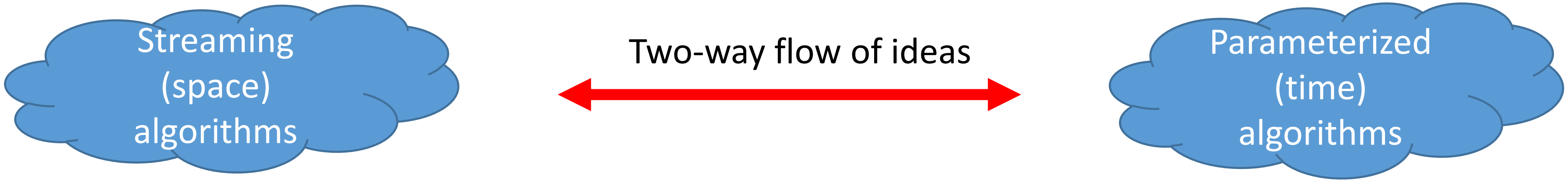
The resulting graph has a triangle iff the edge (y_j, z_ℓ) is present, i.e., I^{th} bit of X is 1

Parameterized Streaming Algorithms

Goal: Develop algorithms and lower bounds to categorize graph problems in this hierarchy

Looking forward…

- The story so far ….
  - Can simulate parameterized techniques (branching, iterative compression, bidimensionality, etc.) in the streaming model
  - Developed new lower bounds using communication complexity
- Beyond "standard" graph problems? Game theory, machine learning, etc …..
- Connections with kernelization?
- Implement and evaluate these new parameterized streaming algorithms?
  - Code for some of the k-VC algorithms available at http://projects.csail.mit.edu/dnd/



Parameterized Streaming Algorithms

Lower bounds inspired by Kernel lower bounds

- Connections with Kernelization – a different (but related) data-compression model
- Kernelization versus streaming
  - Polytime computation versus unbounded computation
  - Full access of the input versus limited access to input
- AND-compression: No poly kernel unless NP⊆ coNP/poly
- New definition of AND-compatible, inspired by AND-compression

A problem Π is AND-compatible if $\exists$ constant $k \in \mathbb{N}$ such that

- $\forall\, n \in \mathbb{N}$ there is a graph G_{YES} on n vertices such that $\Pi(G_{YES}, k)$ is YES instance
- $\forall\, n \in \mathbb{N}$ there is a graph G_{NO} on n vertices such that $\Pi(G_{NO}, k)$ is YES instance
- $\forall\, t \in \mathbb{N}$ we have that $\Pi(G_1 \uplus G_2 \uplus \cdots \uplus G_t, k) = \bigwedge \Pi(G_i, k)$ where $\uplus$ denotes vertex disjoint union

- Many natural graph problems are AND-compatible: k-coloring, k-treewidth, k-girth
- Our result: If a problem Π is AND-compatible then it does not admit a streaming algorithm using space $f(k) \cdot o(n)$, for any function f.
  - Unconditional, unlike kernel lower bounds
- Similar definition and result for OR-compatible

Parameterized Streaming Algorithms

Lower bounds inspired by Kernel lower bounds

A problem Π is AND-compatible if $\exists$ constant $k \in \mathbb{N}$ such that

- $\forall\, n \in \mathbb{N}$ there is a graph G_{YES} on n vertices such that $\Pi(G_{YES}, k)$ is YES instance
- $\forall\, n \in \mathbb{N}$ there is a graph G_{NO} on n vertices such that $\Pi(G_{NO}, k)$ is YES instance
- $\forall\, t \in \mathbb{N}$ we have that $\Pi(G_1 \uplus G_2 \uplus \cdots \uplus G_t, k) = \bigwedge \Pi(G_i, k)$ where $\uplus$ denotes vertex disjoint union

- Our result: If a problem Π is AND-compatible then it does not admit a streaming algorithm using space $f(k) \cdot o(n)$, for any function f.
- Consider t graphs $G_1, G_2, \ldots, G_t$ each having n vertices
- Let G be disjoint union $G_1 \uplus G_2 \uplus \cdots \uplus G_t$
- By pigeonhole principle, any (correct) algorithm for G must use $\geq t$ bits
  - Otherwise two subsets I, J of $[t]$ collide. Let $i^* \in I \setminus J$
  - Select $G_i = G_{YES}$ for each $i \in (I \cup J) \setminus i^*$ and $G_{i^*} = G_{NO}$
  - This violates correctness of the algorithm
- Hence, we have that $f(k) \cdot o(nt) \geq t$
  - Contradiction since k, n are constants and we can take t as large as we want