Graph Stream Model

- **Input:** Sequence of edges \((e_1, e_2 \ldots)\) defines \(n\)-node graph \(G\).

- **Goal:** Compute properties of \(G\) without storing entire graph.

- **Computational constraints:**
  1. Limited working memory, e.g., \(O(n)\) rather than \(O(m)\)
  2. Access data sequentially
  3. Process each element quickly
Motivation

- **Traditional stream applications:** Network monitoring, reading large data sets from disk, aggregation of sensor readings...
- **Interesting theoretical questions:** How can we summarize graphs? Is there a notion of dimensionality reduction? What types of sampling is possible? Connections to compressed sensing, communication complexity, approximation, embeddings, ...
- **Techniques have wider applications:** E.g., distributed settings,

![Diagram of data flow](image)

Each machine runs stream algorithm locally and sends state of their algorithm.
Outline

- **This Talk:**
- **Algorithms:** Summarizing and computing on graph streams
- **Extensions:** Sliding windows, extra passes, annotations etc.
- **Future Directions:** Directed edges, ordering, stochastic graphs
- **Accompanying Survey:**
  - Includes all references and further details.
  - Feedback welcome...

1. Algorithms
2. Extensions
3. Directions
Sparsifiers & Cuts

- **Sparsifiers:** A subgraph $H$ is a $(1 + \varepsilon)$ sparsifier for $G$ if the total weight of any cut is preserved up to a factor $1 + \varepsilon$.

- **Thm:** For any graph $G$ there exists a $(1 + \varepsilon)$ sparsifier with only $O(\varepsilon^{-2} n)$ edges. Can be constructed efficiently.

- **Thm:** Can construct a $(1 + \varepsilon)$-sparsifier of a graph stream using $O(\varepsilon^{-2} n \text{ polylog } n)$ bits of space.
**Algorithm:** Recursively re-sparsify using any “offline” algorithm.

**Analysis:** Let $d=O(\log n)$ be depth of the tree. Error of a final cut estimate is $(1+\varepsilon)^d$ and we only store $d$ sparsifiers simultaneously.

- Results extend to constructing spectral sparsifiers.
Spanners & Distances

- **Spanner**: A subgraph $H$ is a $k$-spanner for $G$ if all graph distances are preserved up to a factor $k$.

- **Thm**: There is a $O(n^{1+1/t})$ space stream algorithm that constructs a $(2t-1)$-spanner.
Spanners Algorithm

- **Algorithm:** Store next edge \((u,v)\) unless it completes a cycle of length \(2t\) or less.

- **Lemma:** All distances preserved up to a factor \(2t-1\) because an edge \((u,v)\) was only ignored if there was already a path of length at most \(2t-1\) between \(u\) and \(v\).

- **Lemma:** At most \(n^{1+1/t}\) edges stored since shortest cycle among stored edges has length at least \(2t+1\).
Other Algorithms

- **Matchings:**
  - **Goal:** Find large set of disjoint edges.
  - **Results:** $\tilde{O}(n)$-space algorithms 2-approx. (unweighted) and 4.91-approx. (weighted). Can do better if edges are grouped together by end-point or arrive in random order.
  - **Extensions:** $O(1)$ approx. for various sub-modular problems.

- **Counting Triangles:** Estimate the number of triangles (or small cycle or clique etc.). See Seshadhri’s talk coming up next...

- **Random Walks:** Simulate length $t$ random walks in $\sqrt{t}$ passes.

- **Other:** Minimum spanning trees, bipartiteness, finding dense components, correlation clustering, independent sets, etc.
1. Algorithms
2. Extensions
3. Directions
Extensions of Model

- **Sliding Window:** Infinite stream but only consider graph defined by recent $w$ edges. Can solve most aforementioned problems.

- **Multiple Passes:** What’s possible with a small number of stream passes? E.g., can find $1 + \varepsilon$ approx. matching in $O(\varepsilon^{-1})$ passes.

- **Annotated Streams:** Suppose a third party “annotates” the stream to assist with the computation. Can we reduce required memory while still verifying correctness.
Dynamic Graphs

- **Dynamic Graph Streams**: Suppose the stream consists of edges both being added and removed from the underlying graph.

- **Can we maintain a uniform edge sample in small space?**
  - Challenge: The sampled edge we have remembered so far may be deleted at the next step.
  - Result: Can maintain uniform sample in $O(\text{polylog } n)$ space via a technique called “$l_0$ sampling”.

- **More powerful sampling techniques:**
  - In $O(n \text{ polylog } n)$ space, can construct a data structure that returns a random edge across any queried cut.
  - In $O(n \text{ polylog } n)$ space, can sample edges where $(u,v)$ is sampled w/p inversely proportional to size of min u-v cut.
Setting: The rows of an adjacency matrix are partitioned between different machines. Equivalently, consider n players each of whom has an “address book” listing their friends.

Goal: Each player sends a “short” message to a third party who then determines if underlying graph is connected.
• **Appears that some messages need to be \( \Omega(n) \) bits:** If there’s a bridge \((u,v)\) in the graph, one of the friends needs to mention this friendship but neither friend knows it’s a bridge.

• **Thm:** \( O(\text{polylog } n) \) bit messages suffice!
  - Protocol is based on dynamic graph sampling results.
  - Also allows third-party to estimate all cut sizes!
1. Algorithms
2. Extensions
3. Directions
Open Problems

Many specific open questions:

- Can we construct a spectral sparsifier in $\tilde{O}(n)$-space with deletions? Best algorithm so far uses $\tilde{O}(n^{5/3})$-space.
- Can we construct spanners of sliding window graphs?
- Improve approx. factors for matchings and triangles...

Open Problems Wiki: Large set of open problems in data streams and property testing can be found at:

http://sublinear.info
Future Directions

Directed Graphs: Almost all research to date has considered undirected graphs but many natural graphs are directed. May need multiple passes but $O(\log n)$ passes might be sufficient.

Stream Ordering: Consider problems under different orderings, e.g., grouped-by-endpoint, increasing weight, random order.

More or Less Space: Most work has focus on $\tilde{O}(n)$-space algorithms. Can we reduce space-complexity for specific families of graphs? What’s possible with slightly more space?

Explore deeper connections with distributed algorithms, communication complexity, dynamic graphs data structures...
Summary of the Survey

• **Algorithms:** Spanners and sparsifiers capture different properties of the graph. Efficient constructions in streaming model. Other positive results for matchings, triangles, etc.

• **Extensions:** Many variants of the basic model including sliding windows, multi-pass, edge deletions, annotations...

• **Directions:** Improve existing results. Future directions include directed graphs, stream ordering, specific graph families etc.

Alice and Bob have $x,y \in \{0,1\}^n$. For Bob to check if $x_i = y_i = 1$ for some $i$ needs $\Omega(n)$ communication.

Let $A$ be an $s$ space algorithm for connectivity.

Consider 2-layer graph $(U,V)$ with $|U| = |V| = n$.

Alice runs $A$ on $E_1 = \{u_iv_i : 1 \leq i \leq n\}$ and $E_2 = \{u_ii_{i+1} : x_i = 0\}$.

Send memory to Bob who runs $A$ on $E_3 = \{v_iv_{i+1} : y_i = 0\}$.

Output of $A$ resolves matrix question so $s = \Omega(n)$. 

---

**Lower Bound for Connectivity**

![Graph Diagram]