Simple Graph Algorithms in the Semi-streaming Model & Map Reduce

Barna Saha
Consider a stream of $m$ edges

\[ \langle e_1, e_2, \ldots, e_m \rangle \]

defining a graph $G$ with nodes $V = [n]$ and $E = \{e_1, \ldots, e_m\}$

- Massive graphs include social networks, web graph, call graphs, etc.
- What can we compute about $G$ in $o(m)$ space?
- Focus on *semi-streaming* space restriction of $O(n \cdot \text{polylog } n)$ bits.
Connectivity

- **Goal:** Compute the number of connected components.
- **Algorithm:** Maintain a spanning forest $F$
  - $F \leftarrow \emptyset$
  - For each edge $(u, v)$, if $u$ and $v$ aren’t connected in $F$,
    \[ F \leftarrow F \cup \{(u, v)\} \]
- **Analysis:**
  - $F$ has the same number of connected components as $G$
  - $F$ has at most $n - 1$ edges.
- **Thm:** Can count connected components in $O(n \log n)$ space.
K-connectivity

- **Goal:** Check if all cuts are of size at least $k$.
- **Algorithm:** Maintain $k$ forests $F_1, \ldots, F_k$
  - $F_1, \ldots, F_k \leftarrow \emptyset$
  - For each edge $(u, v)$, find smallest $i \leq k$ such that $u$ and $v$ aren’t connected in $F_i$,
    $$F_i \leftarrow F_i \cup \{(u, v)\}$$
  - If no such $i$ exists, ignore edge.
- **Analysis:**
  - Each $F_i$ has at most $n - 1$ edges so total edges is $O(nk)$
  - **Lemma:** $\text{Min-Cut}(V, E) < k$ iff $\text{Min-Cut}(V, F_1 \cup \ldots \cup F_k) < k$
  - **Thm:** Can check $k$-connectivity in $O(kn \log n)$ space.
Proof of Lemma

Let $H = (V, F_1 \cup \ldots \cup F_k)$ and let $(S, V \setminus S)$ be an arbitrary cut.

Since $H$ is a subgraph:

$$|E_G(S)| \geq |E_H(S)|$$

where $E_H(S)$ and $E_G(S)$ are the edges across the cut in $H$ and $G$.

Suppose there exists $(u, v) \in E_G(S)$ but $(u, v) \notin F_1 \cup \ldots \cup F_k$. Then $(u, v)$ must be connected in each $F_i$. Since $F_i$ are disjoint,

$$|E_H(S)| \geq \min(|E_G(S)|, k)$$
Minimum Spanning Forest

- **Goal:** Obtain the minimum spanning forest
- **Algorithm:** Maintain a spanning forest $F$
  - Initialize $F \leftarrow \emptyset$
  - Edge $(u,v)$ arrives
    - If $u$ and $v$ not connected in $F$, insert $(u,v)$ in $F$
    - If $u$ and $v$ are connected in $F$ then include $(u,v)$ and find the cycle containing it— remove the edge with minimum weight in that cycle
- **Analysis**
  - $F$ is a forest
  - If an edge $(u,v)$ is not in $F$ then $(u,v)$ must be the heaviest weight edge in some cycle in $G$
- **Thm:** Can maintain minimum spanning tree in $O(n \log(n))$ space
Minimum Spanning Tree in Map Reduce

- Distribute edges randomly to machines. Compute MST on local edges—Combine and Repeat!

- Analysis:
  - Correctness:
    - Use the fact that if an edge is discarded by a machine then it must be the heaviest weight edge in some cycle in a subgraph→ heaviest weight edge in that same cycle in the original graph
    - Hence combine and repeat is a valid policy
  - Complexity
    - Number of rounds required is at most $\left\lceil \frac{C}{\epsilon} \right\rceil$
    - Number of edges before the 1st round $m_1 = n^{1+c}$
    - Number of edges before the 2nd round $m_2 = (n-1)n^{c-\epsilon} = n^{1+c-\epsilon}$ and so on
Minimum Spanning Tree in Map Reduce

• Can we partition the vertices?
  – A more complex algorithm by partitioning the vertices exist with nearly same complexity
  – Works under the same principle of combine & repeat
Graph Streams

• **Sampling Edges**
  – Connectivity, MST, Spanners, Sparsifiers, maximum density estimation....

• **Sampling Vertices**
  – Estimating graph statistic like number of paths of length two/three etc.
Linear Sketch

- **Random linear projection** $M: \mathbb{R}^n \rightarrow \mathbb{R}^k$ that preserves properties of any $v \in \mathbb{R}^n$ with high probability where $k \ll n$.

\[
\begin{pmatrix}
M \\
v
\end{pmatrix}
\begin{pmatrix}
v
\end{pmatrix} = 
\begin{pmatrix}
Mv
\end{pmatrix} \rightarrow \text{answer}
\]

- **Many Results:** Estimating norms, entropy, support size, quantiles, heavy hitters, fitting histograms and polynomials, ...

- **Rich Theory:** Related to compressed sensing and sparse recovery, dimensionality reduction and metric embeddings, ...
Linear Sketch for $F_2$

To construct each row pick a hash function $h:\{1,n\} \to \{+1,-1\}$ uniformly at random from a family of 4-wise independent universal hash family. $z(l,i)=h_i(i)$

Pick $k$ such hash functions independently: $h_1, h_2, \ldots, h_k$ to construct the $k$ rows.
Advantages of Linear Sketch

- Can handle deletion in streams
- Allows for distributed computing

- Exercise: Implement a MapReduce algorithm for computing $F_2$ where the stream is decomposed into $k$ substreams and sent to $k$ different machines initially.

- Similarly there exists linear sketches for graphs to handle deletion of edges.
Sliding/Decaying Window Model

• Only the last $W$ items matter
  – Can you extend the algorithms for Count Min sketch and $F_2$ estimation in the sliding window model?

• Decaying window model
  – No fixed window size but older items have less importance
    • Can you extend the algorithms for Count Min sketch and $F_2$ estimation in the sliding window model?