

# Massive Data Algorithmics

## The Streaming Model

### Lecture 18: Graph Streams

# Maximum Cardinality Matching (MCM)

- The input graph  $G$  is a graph stream.
- Output: A matching with maximum number of edges.

# Algorithm

- **Initialization:**  $M \leftarrow \emptyset$
- **Process**  $(u, v)$ : If  $M \cup \{(u, v)\}$  is a matching,  $M \leftarrow M \cup \{(u, v)\}$
- **Output:**  $|M|$

# Analysis

- Let  $t$  be the output of the algorithm.
- Let  $M^*$  be the maximum matching and  $t^* = |M^*|$ .
- Each edge of  $M$  kills at most two edges of  $M^*$  (preventing to be added to  $M$ ).
- If  $t < t^*/2$ , There exists an unkilld edge in  $M^*$  that could have been added to  $M$ . So  $M$  is not maximal which is a contradiction.
- Therefore,  $t^*/2 \leq t \leq t^*$

# Analysis

- **Definition:** A path whose edges are alternatively in  $M$  and not in  $M$  is called a augmenting path.
- **Theorem:** If there is no augmenting path, then the matching  $M$  is maximum.
- We can find a matching  $M$  such that  $(1 - \epsilon)t^* \leq t \leq t^*$  using constant (depnding on  $\epsilon$ ) number of passes
  - Find a matching in the first pass, and in Passes 2, 3,  $\dots$  find a short augmenting path (depending on  $\epsilon$ ) and increase the size of the matching.

# Maximum Weighted Matching (MWM)

- The input graph  $G$  is a graph stream.
- Output: A matching with the maximum total weight.

# Algorithm

- **Initialization:**  $M \leftarrow \emptyset$
- **Process**  $(u, v)$ : If  $M \cup \{(u, v)\}$  is a matching,  
 $M \leftarrow M \cup \{(u, v)\}$ . Else let  $C = \{\text{edges of } M \text{ conflicting with } (u, v)\}$ . If  $w(u, v) > (1 + \alpha)w(C)$ , then  $M \leftarrow (M - C) \cup \{(u, v)\}$
- **Output:**  $w(M)$

## Analysis

- An edge is born when it is added to  $M$ .
- An edge is died when it is removed from  $M$ . The edge whose inclusion resulted in its removal is called killer.
- An edge survives if it exists in the final  $M$ .
- We can associate a killing tree  $T$  to each survivor edge where survivor is the root.
- If edge  $e$  kills edge  $e'$ ,  $e$  can be seen as the parent of  $e'$ .
- In each killing tree, each node has at most two children.
- Let  $S$  be the set of all survivor edges and let  $T(S)$  be all descendant of the roots of the killing trees.



# Analysis

- We claim that  $w(T(S)) \leq w(S)/\alpha$ .
  - Consider one tree rooted at  $e \in S$ .
  - $w(\text{descendants at level } i) \leq w(\text{descendants at level } i-1)/(1+\alpha)$
  - Therefore,  $w(\text{descendants at level } i) \leq w(e)/(1+\alpha)^i$
  - $w(\text{descendant}) \leq w(e)(\frac{1}{1+\alpha} + \frac{1}{(1+\alpha)^2} + \dots) = w(e)/\alpha$
  - $w(T(S)) = \sum_{e \in S} w(\text{descendant of } e) \leq \sum_{e \in S} w(e)/\alpha = w(S)/\alpha$

# Analysis

- We claim that  $w(M^*) \leq (1 + \alpha)(w(T(S)) + 2w(S))$ .
  - Let  $e_1^*, e_2^*, \dots$  be the edges in  $M^*$  in the stream order.
  - If  $e_i^*$  is born, charge  $w(e_i^*)$  to  $e_i^*$  which is in  $T(S) \cup S$ .
  - If  $e_i^*$  is not born, this is because of 1 or 2 conflicting edges
    - One conflicting edge  $e$ : note  $e \in S \cup T(S)$ . Charge  $w(e_i^*)$  to  $e$  (indeed we charge  $w(e_i^*)$  to the common endpoint of  $e$  and  $e_i^*$ ). Since  $e_i^*$  could not kill  $e$ ,  $w(e_i^*) \leq (1 + \alpha)w(e)$ .
    - Two conflicting edges  $e_1$  and  $e_2$ : Note  $e_1, e_2 \in S \cup T(S)$ . Charge  $w(e_i^*) \cdot \frac{w(e_j)}{w(e_1) + w(e_2)}$  to  $e_j$  for  $j = 1, 2$  (again we charge to the common endpoint of  $e_j$  and  $e_i^*$ ). Since  $e_i^*$  could not kill  $e_1, e_2$ ,  $w(e_i^*) \leq (1 + \alpha)(w(e_1) + w(e_2))$ . As before, we maintain the property that weight charged to an edge  $e \leq (1 + \alpha)w(e)$ .
  - If an edge  $e$  is killed by  $e'$ , transfer charge assigned to the common endpoint of  $e$  and  $e'$  from  $e$  to  $e'$ .
  - To each edge of  $T(s)$ , at most one charge remains (one is transferred to its parent) but each edge in  $S$  is charged twice.

# Analysis

- Combining two claims

$$w(M^*) \leq (1 + \alpha)(w(S)/\alpha + 2w(S)) = \left(\frac{1}{\alpha} + 3 + 2\alpha\right)w(S)$$

- .
- Best choice for  $\alpha$  minimizing the above expression is  $\frac{1}{\sqrt{2}}$ .
- This gives us:

$$\frac{w(M^*)}{3 + 2\sqrt{2}} \leq w(M) \leq w(M^*)$$

# Triangle Counting

- The input graph  $G$  is a graph stream.
- Output: A estimation of the number of triangles

## Some Known Results

- We can not multiplicatively approximate the number of triangles in  $o(n^2)$  space.
- We can approximate the number of triangles up to some additive error.
- If we are given that the number of triangles  $\geq t$ , then we can multiplicatively approximate it.

# The First Algorithm

- Pick a random edge  $(u, v)$  u.a.r. from the stream.
- Pick a vertex  $w$  u.a.r. from  $V - \{u, v\}$ .
- If  $(u, w)$  and  $(v, w)$  appears after  $(u, v)$  in the stream, output  $m(n-2)$  else output 0.

## Intuition and Analysis

- The expectation of the output is the number of triangles.
- Run several copies of the algorithms in parallel and take the average of their output to be the answer.
- Using Chebyshev's inequality, from the variance bound, we get the space usage to be  $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \cdot (\frac{mn}{t})^2)$

## The Second Algorithm

- Produce from the actual token  $\{u, v\}$ , the virtual tokens  $\{u, v, w_1\}, \{u, v, w_2\}, \dots, \{u, v, w_{n-2}\}$  ( $w_i \in V - \{u, v\}$ )
- Let  $F_k$  be the  $k$ -th frequency moment of the virtual stream.



## Intuition and Analysis

- Let  $T_i = |\{\{u, v, w\} : u, v, w \text{ are distinct vertices and } \exists \text{ exactly } i \text{ edges among } \{u, v, w\}\}|$
- We know  $T_0 + T_1 + T_2 + T_3 = C_3^n$
- $F_2 = \sum_{u,v,w} (\text{number of occurrences of } \{u, v, w\} \text{ in the virtual stream})^2 = 1^2 \cdot T_1 + 2^2 \cdot T_2 + 3^2 \cdot T_3 = T_1 + 4T_2 + 9T_3$
- Similarly,  $F_1 = T_1 + 2T_2 + 3T_3$ . On the other hand,  $F_1 = m(n-2)$ , the length of the virtual stream.
- $F_0 = T_1 + T_2 + T_3$
- If we had estimates for  $F_0, F_1$  and  $F_2$ , we could compute  $T_3$  by solving the above equations.
- So, we need to compute two sketches of the virtual stream, one to estimate  $F_0$  and the other to estimate  $F_2$ .

## References

- **Data Stream Algorithms** (Chapter 14)  
Lecture notes by A. Chakrabbarti and D. College