CMPUT 675: Algorithms for Streaming and Big Data **Fall 2019**

Lecture 14 (Oct 23, 2019): Matching and Connectivity

Lecturer: Mohammad R. Salavatipour *Scribe: Candelario Gutierrez*

14.1 Matching

A graph M is said to be a matching if each vertex has a max degree 1, in other words no two edges have a common end-point.

14.1.1 Problem

Given an unweighted or weighted graph (in streaming model) we want to find a maximum matching. There are two types of the problem depending on the graph:

- Maximum Cardinality Matching (MCM): Find the matching with the largest size (in unweighted graphs).
- Maximum Weighted Matching (MWM): Find the matching with the maximum sum of weights (in weighted graphs).

14.1.2 Simple Offline 2-approximation for MCM

In general, in the off-line setting, a simple 2-approximation for MCM is using the fact that any *maximal* matching is a good approximation of the maximum matching. In the streaming model, we can convert it into a one-pass algorithm that finds a maximal matching $\hat{M}$ (a 2-approximation for the maximum matching) was proposed in 2005 by Feigenbaum, Kannan, McGregor, Suri, and Zhang [FKMSZ05]:

MCM

$M \leftarrow \emptyset$

For each edge e do
  if $M \cup \{e\}$ is a matching
    $M \leftarrow M \cup \{e\}$
return M

This can be adapted to MWM through its parametrization.

14.1.3 MWM

MWM

$M \leftarrow \emptyset$

For each edge $e = (u, v)$ in the stream do
  if $M \cup \{e\}$ is a matching item
    $M \leftarrow M \cup \{e\}$
  else let C be the set of edges conflicting with e
    if $W_e > (1 + \alpha)W_C$ then
      $M \leftarrow (M - C) \cup \{e\}$
return M

We will show the following:

Theorem 1 *This is a* $(3 + 2\sqrt{2})$*-approximation for MWM in space* $O(n \log n)$, $\widetilde{O}(n)$.

Here are some definitions we use through our analysis.

- We say edge e is born when we add it to M.
- We say it died (or was killed) when removed.
- Each dead edge has a killer.
- Survives if it stays in M until the end.
- We build a tree associated with killer/survivor relation.

For survived edge e, T_e rooted at e has ≤ 2 children nodes, the ones that e killed when born. The subtrees of the children nodes are defined recursively:

$$S = \{survivors\} \text{ and } T(S) = \bigcup_{e \epsilon S} \{ \text{ edges of killer tree of } e\}.$$

Lemma 1
- $W(T(S)) \leq \frac{W(S)}{\alpha}$
- *optimum (opt)* $\leq (1 + \alpha)(W(T(S)) + 2W(S))$

Proof. Consider killer tree of one $e \in S$:

$$\begin{aligned} W(\text{level } i \text{ descendents of } e) &\leq \frac{W(level\ i-1)}{1+\alpha} \\ &\leq \frac{W(e)}{(1+\alpha)^i} \end{aligned}$$

$$\begin{aligned} W(T_e) \leq & W(e)(\frac{1}{1+\alpha} + \frac{1}{(1+\alpha)^2} + ...) \\ = & \frac{W(e)}{1+\alpha}(\frac{1}{1-\frac{1}{1+\alpha}}) \\ = & \frac{W(e)}{1+\alpha-1} \\ = & \frac{W(e)}{\alpha} \end{aligned}$$

Let $e_1^* e_2^* ... e_m^*$, be the edges of an opt matching M^* in the order they appear in the stream.

We define the following charging:

- If e_i^* is born then charge $W(e_i^*)$ to e_i^* in $S \cup T(S)$
- If e_i^* is not born (because of conflicts):
  - One conflict edge e: $e \in S$
    - Charge $W(e_i^*)$ to e
    - $W(e_i^*) \leq (1+\alpha)W(e)$ (because it didn't kill e)
  - Two conflicting edges e_1,e_2:
    - Charge e_1: $W(e_i^*)\frac{W(e_1)}{W(e_1)+W(e_2)}$
    - Charge e_2: $W(e_i^*)\frac{W(e_2)}{W(e_1)+W(e_2)}$
    - $W(e_i^*) \leq (1+\alpha)(W(e_1)+W(e_2))$
- If an edge e is killed by e' we transfer charge to the killer.
  - $W(e) \leq W(e')$
  - We maintain that weight charged to an edge e is $\leq (1+\alpha)W(e)$
- Edges e in S may have to absorb $2(1+\alpha)W(e)$ charge because of up to two conflicts with edges in M^*:

$$\begin{aligned} opt \leq & (1+\alpha)(\frac{W(S)}{\alpha} + 2W(S)) \\ = & (\frac{1}{\alpha} + 3 + 2\alpha)W(S) \end{aligned}$$

Choose $\alpha = \frac{1}{\sqrt{2}} \rightarrow (3+2\sqrt{2})$-approximation.

■

14.2 Connectivity in dynamic (insert/delete) model

We revisit the problem of connectivity (or finding connected components) int he dynamic graph model where edges are added/deleted in the stream. One can think of the following algorithm (in the off-line setting) to

find connected components. We start with each vertex being a singleton component. At each iteration we pair up the components that have an edge between them and merge them into one big super-node. We repeat this until there are no edges left between the supernodes or there is only one supernode left. The super-nodes left represent the connected components of the graph. If at each round the nodes that merge represent roughly equal size components, then the size of each component doubles after each merger and hence we have $O(\log n)$ iterations.

For each $v \in V$, let $x^v \in \{-1, 0, 1\}^{\binom{n}{2}}$:

$$x^v(u,w) \in E = \begin{cases} 1 & if\ u = v,\ u < w \\ -1 & if\ u = v,\ u > w \\ 0 & Otherwise \end{cases}$$

If S is the set of vertices in a super node, the non-zero entries of $\sum_{v \epsilon S} x^v$ are exactly the edges comming out of $S^{v \epsilon S}$. The algorithm for connectivity in the dynamic model has the following structure. We keep sketches for $x^v s$. When we see an edge (u, v) we update $x^v(u,v) = \begin{cases} 1 & if\ u < v \\ -1 & if\ u > v \end{cases}$ (Similarly for $x^u(u,v)$) When we have deletion we do update with reverse sign.

So we have sketches for $x^v s$. We can do ℓ_0 sampling; i.e. sample a random edge incident.

For a connected subgraph S let

$$x^S = \sum_{v \in S} x^v$$

$$x^S_{(u,v)} \begin{cases} 1 & if\ u \epsilon S,\ v \notin S,\ u < v \\ -1 & if\ u \epsilon S,\ v \notin S,\ u > v \\ 0 & Otherwise \end{cases}$$

Note that for (u, v), if $u, v \in S$ then x^u, x^v will have different signs and cancel out each other in the sum.

By linearity of the sketches,

$$\ell_0\ sketch\ of\ S = \sum_{v \epsilon S} (\ell_0 - sketch\ of\ x^v)$$

The algorithm maintains sketches for $x^v s$. So the algorithm maintains ℓ_0-sketches (x^v).

14.3 Estimation Algorithm

- Maintain connected components (super nodes) $S_1 ... S_n$.
- For each S_i sample an edge using ℓ_0 sampling.
- Update the connected components by combining if there is a sampled edge between them.
- Repeat until 1 connected component or no more edges.

Expected number of rounds is $O(\log n)$ and a total space of $O(n\ polylog\ n)$.

References

M14 A. Mcgregor, Graph Stream Algorithms: A Survey. *SIGMOD Record*, 43(1):9-20, 2014.

FKMSZ05 J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang, On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2):207–216, 2005.