University of Wrocław: Algorithms for Big Data (Fall'19) 20/01/2020

# Lecture 13: MPC

Lecturer: *Przemysław Uznański* Scribe: -

## 1 Massively Parallel Computing [LMSV11]

Modern distributed computing model (captures map-reduce, hadoop, spark, etc.).

- input of size $n$
- $k$ machines, each of space $S$, $S = n^{1-\delta}$, $k \cdot S = \mathcal{O}(n)$.
- output (might be too large, e.g. size $n$ that does not fit on a single machine)

Computation:

- computation happens over $R$ rounds
- each machine: near linear computation per round, so total computation cost $\mathcal{O}(n^{1+o(1)}R)$
- each machine communicates $\sim S$ bits per round, so total communication cost $\mathcal{O}(nR)$

goal: minimize $R$

### 1.1 Sorting (Tera-Sort)

Intuition: if we partition input onto machines, so each machine receives contiguous fragment of size $S$, then we are done in a single round (each machine sorts and outputs its own part, output == concatenation of outputs).

Idea:

- each machine receives input
- each machine samples randomly from its input
- each sample is sent to single machine (1 round)
- 1 machine gathers all the samples, sorts locally, and sents back to everyone approximate histogram
- machines use approximate histogram to decide how to partition locally their input and sent it to proper receivers
- then everyone sorts their parts

Total sample size $s = \mathcal{O}(S)$, and whp histogram is with $\pm\varepsilon n$ error, where $\frac{\log n}{\varepsilon^2} \leq s$. We are ok if error is $\mathcal{O}(S)$. This is satisfied when $S^{3/2} = \Omega(n\sqrt{\log n})$, or $S = \widetilde{\Omega}(n^{2/3})$, or $k = \widetilde{\mathcal{O}}(n^{1/3}) \implies$ sorting in $\mathcal{O}(1)$ rounds

## 2 Connectivity

Input: list of edges, partitioned (arbitrarily), size $N$ output: each vertex labeled with id of connected component

Notation: id of vertex $\pi(v)$, label of vertex $\ell(v)$, $\Gamma(v)$ denotes neighbourhood Approach:

- let $L_v$ be the set of vertices $\{u : \ell(u) = \ell(v)\}$ – always a subset of connected component of $v$
- initially $\ell(v) \leftarrow \pi(v)$
- some vertcies are called *active*
- every $L_v$ will have exactly one active vertex (e.g. one with smallest $\ell$), that makes decisions wrt whole $L_v$, assume cannonically its $v$
- in each round:
  - each active vertex becomes a *leader* with ppb $1/2$
  - if $v$ is a leader, mark whole $L_v$ as a leaders
  - every active non-leader $v$, find $w \in \Gamma(L_v)$ that is a leader and minimizes $\pi(w)$
  - whole $L_v$ joins $L_w$

Observation: each round decreases *in expectation* number of components by a factor of $1/4$, so $\mathcal{O}(\log n)$ rounds are enough. Implementation details:

- edges are stored locally
- split vertices of large degree into smaller vertices of degree $\mathcal{O}(S)$ (this is already non-trivial and requires e.g. sorting)
- ditributed data structure implementing vertex info required (implemented e.g. via sorting, emulates sending messages over the edges etc.)

## 3 MST

Input: graph of $N = n^{1+\delta}$ edges. Output: some approximation of a maximal matching. Output fits into single machine (one edge per vertex). Assume $S = n^{1+\varepsilon}$.

- Split output randomly into machines.
- Each machine $i$ computes MST of its input $T_i$
- Recurse on $\bigcup_i T_i$

Filtering step reduces total number of edges from $n^{1+\delta}$ to $n^{1+\delta-\varepsilon}$, thus the number of rounds is $\mathcal{O}(\delta/\varepsilon)$.

# 4 Maximal matching

(Setting the same as in previous algorithm)

- Sample $E'$, set of edges of size $\mathcal{O}(S)$.
- Send $E'$ to one machine. Compute $M'$, the maximal matching of $E'$.
- Remove all vertices from $V[E']$ from graph. Recurse on remaining graph. Call $M$ the result of recursion.
- Return $M \cup M'$.

Correctness: simulated greedy.

**Lemma 1.** *If $E'$ is a set of edges picked by sampling with probability $p$. Let $I$ be set of vertices not adjacent to $E'$. With very high probability, $|E[I]| \leq 2n/p$.*

*Proof.* Follows from concentration bounds, since if we pick set $E_1$ of size $2n/p$, then expected intersection size $\mathbb{E}|E[I] \cap E'| = 2n$, so with exponentially small probability its not empty. We then take union bound over all possible subsets. □

**Theorem 2.** *Number of rounds is $\mathcal{O}(\delta/\varepsilon)$.*

*Proof.* Sampling probability needs to be $p = S/m = n^{\varepsilon-\delta}$. So in each step number of edges is reduced from $n^{1+\delta}$ to $2n^{1+\delta-\varepsilon}$. □

# 5 Maximal matching, approach 2

- Partition edges randomly into machines.
- Each machine receives $G_i$.
- Each machine computes $M_i$, maximal matching of $G_i$.
- Everyone sends $M_i$ to a coordinator machine.
- Coordinator outputs $M$, maximal matching from $\bigcup_i M_i$.

The limiting factor is that all of the maximal-matchings need to be aggregated on a single machine, so $S \geq n \cdot \frac{N}{S}$, or $S \geq \sqrt{n \cdot N}$, which for dense graphs is $n^{3/2}$.

Claim: algorithm outputs maximal matching of size $\Theta(n)$, so its $\Theta(1)$-approximation of MM.

*Proof.* Consider greedy algorithm going through edges in order of $M_1, \ldots, M_k$. W.l.o.g. maximal matching is of size $\mathcal{O}(n)$. Consider step $i$ (processing $M_i$).

**Lemma 3.** *Assume we have already selected $o(n)$ edges from $M_1 \cup \ldots \cup M_{i-1}$, as otherwise we are done.*
*$E_{old}$ – all edges in $G_i$ that are adjacent to already selected vertex*
*$\mu_{old}$ – size of a maximum matching in $G_i$ using only edges in $E_{old}$.*
*There exists matching in $G_1 \cup \ldots \cup G_i$ of size $\mu_{old} + \Omega(n/k)$.*

*Proof.* $G$ contains a matching of size $\Theta(n) - 2\mu_{\text{old}} = \Theta(n)$ that is not adjacent to any of the vertices from already selected maching. By random partitioning, $\Theta(n/k)$ of those edges land in $G_i$, so $G_i$ contains matching of size $\mu_{\text{old}} + \Omega(n/k)$. □

So any maximal matching in $G_i$ needs to be of size at least $\mu_{\text{old}} + \Omega(n/k)$. At most $\mu_{\text{old}}$ of edges in any maximal matching can be adjacent to vertices already blocked in previous rounds, so from maximal matching we are getting $\Omega(n/k)$ new edges. □

## References


[LMSV11] Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: a method for solving graph problems in mapreduce. In Rajmohan Rajaraman and Friedhelm Meyer auf der Heide, editors, *SPAA 2011: Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures, San Jose, CA, USA, June 4-6, 2011 (Co-located with FCRC 2011)*, pages 85–94. ACM, 2011.