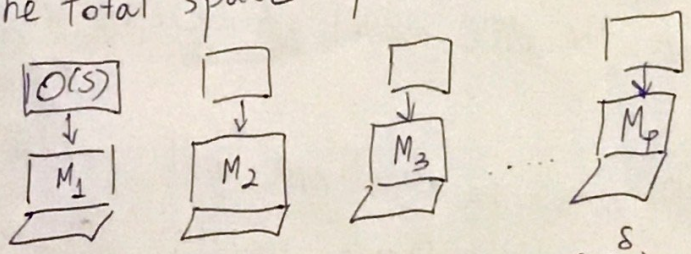
Parallel Graph Connectivity in Log Diameter Rounds

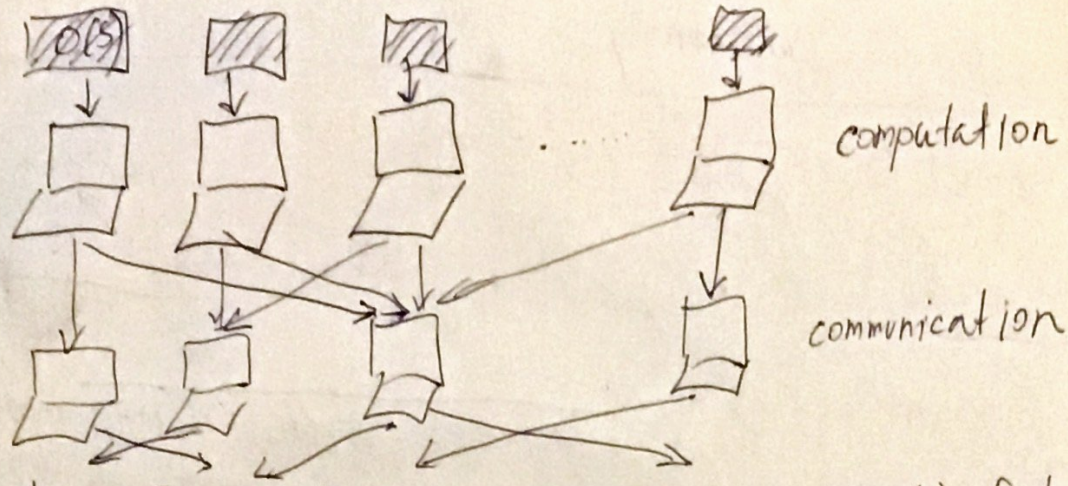
[Andoni-Song-Stein-Wang-Zhong FOCS'18]

- Main result: A $O(\log D \log\log_{m/n})$-round graph connectivity (identifying connected components (CCs)) algorithm on the Massive Parallel Computation model (MPC model) for graphs with n nodes, m edges, and diameter D, using $\Theta(m)$ total memory under the strongly sublinear memory regime.

- The MPC model:
  - Computation model for several parallel systems such as MapReduce, Hadoop, Spark and etc.
    [Karloff-Suri-Vassilvitskii '10, Beame-Koutris-Suciu '13]
  - In the model:
    - There are p machines, each with local space S.
    - The input data of size N is arbitrarily partitioned into these machines, so each machine holds a part of input data of size $O(S)$
    - The total space $p \cdot S = O(N)$



  - Truly sublinear regime: $S = O(N^{\delta})$ for some constant $\delta \in (0,1)$
  - Computation of MPC proceeds in rounds:
    - In each round, each machine can do local computation based on data stored on local memory, and can send messages to arbitrary other machine.



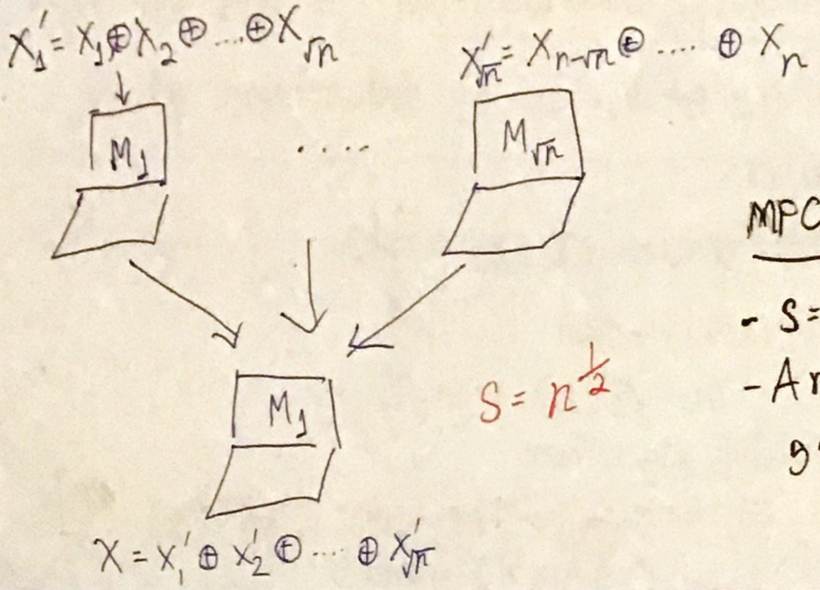
- Restriction: Each machine can send/receive message(s) of total size $O(s)$.

- The output of the computation is distributed on output machine(s).

- Goal: Design algorithms that minimize the # of rounds.

- MPC vs PRAM:

- The MPC model captures the power of "local computation" while the classical parallel RAM (PRAM) model captures the power of "data pallelism".
- Good news: We can reuse any PRAM algorithm on the MPC model.
  - ▷ Any PRAM algorithm with parallel time $t(n)$ can be simulated on the MPC model in $O(t(n))$ rounds [KSV '10]
  - ▷ Typically, the simulating algorithm will perform in logarithmic / poly logarithmic rounds. [Beame-Hastad '89]
    - XOR of n bits require $\tilde{\Omega}(\log n)$ on a CRCW-PRAM.
    - XOR of n bits on the MPC model can be done in $O(\log_s n) = O(1)$ [Beame-Hastad '89]



$S = n^{\frac{1}{2}}$

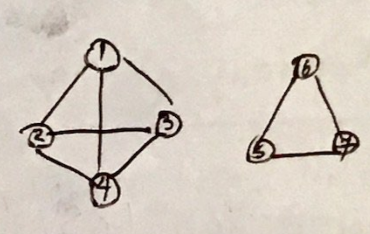
MPC as a circuit:

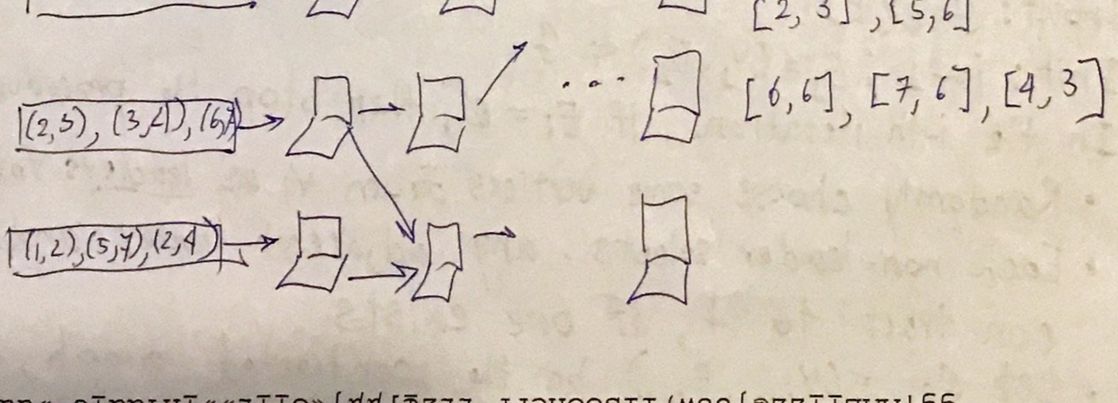
- $S = n^{\delta}$ fan-in gates
- Arbitrary function at a gate.

Researsh ~~goal~~ challenge: For which problems can we design MPC algorithm that perform faster than the best PRAM algorithm?

- Graph Connectivity in the MPC model
  - Input: Edges of an n-vertex, m-edge graph $G=(V,E)$
  - Output: Connected components of G
    - (vertex, color) pairs: each vertex is assigned a color.
    - Vertices v and v' are in the same CC iff v and v' are assigned by the same color.

Ex





- Previous Work until [ASSWZ '18]

  - Total space $N = m$ (# of edges)
  - n denotes # of vertices
  - Local space $S = \Theta(n^{1+\epsilon})$ in $O(1)$ rounds
    [Lattanzi-Moseley-Suri-Vassilvitskii '11]
  - Local space $S = \Theta(n)$ in $O(1)$ rounds
    - Simulate CONGEST-CLIQUE algorithm
      [Jurdzinski-Nowicki '17, Behnezhad-Derakhshan-Hajiaghayi '18]
  - Local space $S = \Theta(n^{\delta})$ in $O(\log n)$ rounds
    - Simulate a $O(\log n)$-parallel time PRAM algorithm
      [Shiloach-Vishkin '82]
    - A randomized algorithm in $O(\log n)$ rounds
      [KSV '10]

Main question: Can we design an MPC graph connectivity algorithm that is faster than $O(\log n)$ rounds with local space $s < n$?

Main result:

- An MPC graph connectivity algorithm in $O(\log D \cdot \log\log_{m/n} n)$ rounds with local space $S = n^{\delta}$ where $\delta \in (0,1)$ and total space $p \cdot S = \tilde{\Theta}(m)$.

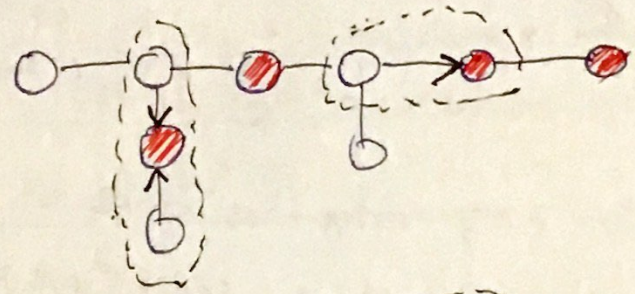
Algorithmic Framework:

Leader-contraction Algorithm →

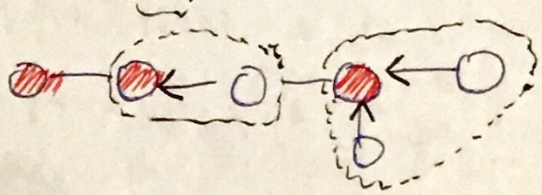
- Based on the idea of "leader-contraction" [KSV '10]
  - Input: $G = (V, E)$
  - Init: $i \leftarrow 1$; $G_1 = (V_1, E_1) \leftarrow G$
  - In the i-th iteration, if $E_i = \emptyset$, then stop the procedure
    - Randomly choose some vertices from V_i as leaders for G_i
    - Each non-leader selects any adjacent leader and contract to it, if one exists
    - Let $G_{i+1} = (V_{i+1}, E_{i+1})$ be the contracted graph, and set $i \leftarrow i+1$

- At the end of the procedure, for each $v \in V$, if v is finally contracted to u, then assign v the color u.

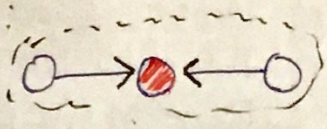
Ex G_1:



G_2:



G_3:



G_4: ○

- Analysis of Leader-Contract Algorithm (warm-up 1)
  - Not hard to see that at the end of the procedure, each surviving vertex corresponds to a CC.
  - If the leaders for G_i are already chosen, then the contracted graph G_{i+1} can be constructed in $O(1)$ rounds [KSV'10]
  - Suppose probability of a vertex being a leader is $\frac{1}{2}$
  - # of vertices in G_{i+1} = # leaders in G_i + # of non-leader which has no adjacent leader in G_i

$\Rightarrow \Pr\{\text{a node being non-leader w/ no adjacent leader}\} = \frac{1}{2} \times \left[\left(\frac{1}{2}\right)^{\deg(v)}\right] \leq \frac{1}{4}$, as $\deg(v) \geq 1$

  - Hence, in expectation, # of vertices in $G_{i+1} \leq \frac{3}{4}$ (# of vertices in G_i).
  - Therefore, the algorithm will terminated in $O(\log n)$ expected rounds (optimal for path graphs)

- Leader Selection For High Degree Graphs

- Idea to reduce round complexity [ASSWZ '18]
  - To improve round complexity $\ll \log n$, we should choose a much smaller fraction of the vertices to be leaders.

- For a graph where every vertex has degree at least $d \gg \log n$, we can choose each vertex as a leader w/ probability $\Theta(\frac{\log n}{d})$

  $\Rightarrow$ # of leaders will be $\tilde{O}(n/d)$

  $\Rightarrow$ Every non-leader will have at least 1 adjacent leader

  $\Rightarrow$ After each contraction, only $\tilde{O}(\frac{1}{d})$ fraction of vertices will survive.

- Approach by [ASSWZ '18]:

  Convert to high-degree graph + leader-contraction (Densification)

- Densification:
  - Convert an input graph $G \rightarrow$ a new graph G' s.t.
    - The vertices of G = the vertices of G'
    - Each vertex in G' has degree at least d
    - G' preserves the connectivity of G.

Densification via Graph Exponentiation:

- A simple algorithm that connects every vertex to vertices within its 2-hop by adding edges.

  $\Rightarrow$ The distance between any two vertices shrinks by a factor of 2.

  Not hard to see that $\Rightarrow$ Each CC becomes a clique in $O(\log D)$ steps

Problem: Total space required can be up to $\Omega(n^2)$ which for sparse graphs is much larger than $O(m)$.

- [ASSWZ '18] manages to improve the total space to $O(m)$ at the cost of increasing rand complexity to $O(\log D \log\log_{m/n} n)$

Graph-connectivity Algorithm →

The Proposed Algorithm [ASSWZ '18]

- Input: $G=(V,E)$
- Init: $i \leftarrow 1$; $G_1=(V_1,E_1) \leftarrow G$
- In the i-th iteration, if $E_i=\emptyset$, then stop the procedure
  - Convert G_i into $G_i'=(V_i,E_i')$ with mindegree $\geq d_i$
  - Densification → Randomly choose some vertices from V_i as leaders for G_i with probability $\min\{\frac{1}{2}, \frac{\log n}{d_i}\}$
  - Each non-leader selects any adjacent leader and contracts into it, if one exists
  - Let $G_{i+1}=(V_{i+1},E_{i+1})$ be the contracted graph, and set $i \leftarrow i+1$
  - At the end of the procedure, Foreach $v \in V$, if v is finally contracted to u, then assign v the color u.

Densification Lemma: Graph G_i with n nodes can be converted into G_i' in $O(\log D)$ rounds and the total number of extra added edges is $O(nd^2)$.

$G_1 \xrightarrow{\text{Densification (increase \# of edges)}} G_1' \Rightarrow G_2 \xrightarrow{\text{Densification}} G_2' \Rightarrow G_3 \Rightarrow \cdots$

$G_1' \Rightarrow G_2$: leader-contraction (decrease \# of nodes)

$G_2' \Rightarrow G_3$: leader-contraction

Analysis of Graph-Connectivity-Algorithm:

- In iteration i, graph G_i has n_i nodes
- With total space $O(m)$, we can afford densification with $d_i = \sqrt{\frac{m}{n_i}}$
- Leader contraction reduces to $n_{i+1} = \tilde{O}(\frac{n_i}{d_i})$

$$n_{i+1} = \tilde{O}\left(\frac{n_i^{1.5}}{m^{0.5}}\right)$$

$$= \tilde{O}\left(\frac{1}{m^{0.5}} \times \left(\frac{n_{i-1}^{1.5}}{m^{0.5}}\right)^{1.5}\right)$$

$$= \dots$$

$$= \tilde{O}\left(\frac{n^{1.5^i}}{m^{0.5^{i-1}}}\right) = \tilde{O}\left(n \times \left(\frac{n}{m}\right)^{1.5^i - 1}\right)$$

$$= \tilde{O}\left(\frac{n}{\left(\frac{m}{n}\right)^{1.5^i - 1}}\right)$$

- If $i = \Omega(\log \log_{m/n} n)$, then $n_i = O(1)$
- Therefore, the total # of rounds is $O(\log D \cdot \log \log_{m/n} n)$.

Extended Results and Application:

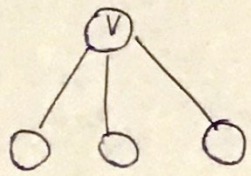
- Finding spanning forest in $O(\log D \cdot \log \log_m n)$ rounds
- Finding a DFS sequence for a forest
- Exact / approximate MST

Recent Advance

- Improvement to $O(\log D)$ rounds if $D > \log^6 n$ and $O(\log \log n)$ rounds otherwise.

Local operations for Leader-Contraction

- One iteration of Leader-Contraction:
  - Each vertex v and its incident edges assigned to a handler machine
  - Handler machine for v:



    - Determines whether v is leader, then communicates to handlers of neighbors
    - If v is not a leader, select a unique adjacent leader, sends incident edges of v to that leader, stores that v in the same CC w/ the leader
    - Issue: What if degree $d \geq S$?
      ⇒ Use d/S handlers and process via a tree of depth $\lceil \log_S d \rceil = O(1)$
      $(\log_{n^\delta} n^{\delta \frac{1}{\delta}} = \frac{1}{\delta} \log_n n^\delta = O(1))$
  - Assignment to handlers
    - random
    - load balancing

Rule-of-thumb: Most "simple" local operations can be done in $O(1)$ round, as long as total space bound is satisfied.

??I'±Fé¼± ± «Ÿíôô`ôôzb
$>|¥ññòÌ-- ±««± »þ££u9»""hŸŸÛ»»âQ¢TÜBgÄÑÑ ± »

??I'±Fé¼± ± «Ÿíôô`ôôzb
$>|¥ññòÌ-- ±««± »þ££u9»""hŸŸÛ»»âQ¢TÜBgÄÑÑ ± »

þ±±¤- »aõõ■Ãjj[[Ê¡¡±kqqçÚÚ°cc

þ±±¤- ± 1>ï±±ÿ}lzoo<

Densification Algorithm

- Densification Lemma (recall): Converting can be done in $O(\log D)$ rounds and $O(nd^2)$ total space (# of total extra edges)

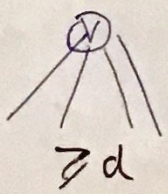
- Idea: "truncated broad casting"
  - Let $T(v)$ denote the set of neighbors of v.

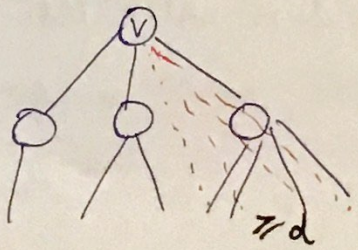
Densification-Algorithm

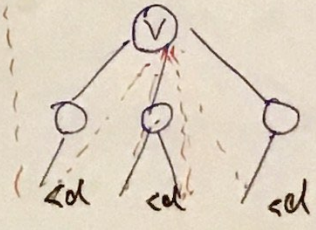
- Init: $i \leftarrow 1$; $\forall v \in V$, $T_1(v) \leftarrow T(v) \cup \{v\}$
- In the i-th round, if $\forall v \in V$, $|T_i(v)| \geq d$, stop the procedure
- For each $v \in V$,
  (1) if $|T_i(v)| \geq d$, then $T_{i+1}(v) \leftarrow T_i(v)$
  (2) else if $\exists u \in T_i(v)$ and $|T_i(u)| \geq d$, then $T_{i+1}(v) \leftarrow T_i(u) \cup T_i(v)$
  (3) else $T_{i+1}(v) \leftarrow \bigcup_{u \in T_i(v)} T_i(u)$

Analysis of Densification-Algorithm

- Case (1) $\Rightarrow$ space $O(1)$ | Case (2) $\Rightarrow$ space $O(d)$ | Case (3) $\Rightarrow$ space $O(d^2)$







- $T_{i+1}(v)$ contains all the vertices which have distance to v at most 2^i (by induction) $\Rightarrow O(\log D)$ rounds