

# Distributed Algorithms for Connectivity and MST in Large Graphs with Efficient Local Computation

Eric Ajieren, Khalid Hourani, William K. Moses Jr., Gopal Pandurangan

University of Houston

January 4, 2022

# Table of Contents

- 1 Introduction
- 2 Parallel Flooding
- 3 Filtering-Based MST
- 4 Deterministic Borøuvka-Style Algorithm
- 5 An Almost-Optimal Randomized Algorithm

# Table of Contents

- 1 Introduction
- 2 Parallel Flooding
- 3 Filtering-Based MST
- 4 Deterministic Borůvka-Style Algorithm
- 5 An Almost-Optimal Randomized Algorithm

# The $k$ -Machine Model

- study distributed algorithms for large-scale graphs
- focus on connectivity and MST
- in  $k$ -machine model
  - $k \geq 2$  machines jointly perform computations on input graph
  - input graph distributed uniformly at random to  $k$  machines
    - $n$  nodes,  $m$  edges
    - when a node is given to a machine, its **adjacency list** is also
    - commonly called **vertex centric** model
  - assume  $n \gg k$  (e.g.  $k = \mathcal{O}(n^\epsilon)$ )
- typically, **only consider communication cost**
  - message complexity — number of communication rounds
- traditionally done because network communication is much slower than local computation
  - less true as network speeds increase

# Our Results

- we posit a new complexity measure
  - **local computation cost** —  $T_\ell$
  - measures the worst-case local computation cost among  $k$  machines
- refer to traditional communication complexity by  $T_c$
- lower bounds

$$T_\ell = \Omega\left(\frac{m+n}{k} + \Delta + k\right)$$

$$T_c = \Omega\left(\frac{n}{k^2}\right)$$

- in practical implementations of  $k$ -machine model algorithms, often see suboptimal speedup
  - e.g. PageRank with  $T_c = \mathcal{O}(n/k)$  but whose wall-clock time is approximately  $n/k^{0.8}$

- analyze several algorithms under the new complexity measure

Algorithm	Round complexity	Local runtime
Flooding	$\tilde{O}\left(\frac{n}{k} + D\right)$	$\tilde{O}\left(\frac{m}{k} + \Delta + k\right)$
Filtering	$\tilde{O}\left(\frac{n}{k}\right)$	$\tilde{O}\left(\frac{m}{k} + n\right)$
Improved Local Borůvka	$\tilde{O}\left(\frac{n}{k}\right)$	$\tilde{O}\left(\frac{m+n}{k} + \Delta + k\right)$
Randomized CC	$\tilde{O}\left(\frac{n}{k^2}\right)$	$\tilde{O}\left(\frac{m+n}{k} + \Delta + k\right)$

- many distributed algorithms that are optimal are often only optimal under **communication complexity**
- as a byproduct of our analysis, we have two general results
  - the Node Distribution Lemma
  - the Mapping Lemma

## Lemma (Node Distribution Lemma)

*Consider a graph  $G$  of nodes  $v_1, v_2, \dots, v_n$  with associated non-negative real-valued “weights”  $w(v)$  for each node  $v$ . Given a uniform, random distribution of the  $n$  nodes to  $k$  machines, as in the  $k$ -machine model, then, with probability at least  $1 - 1/n^a$  for any  $a > 0$ , the total weight of nodes at every machine is bounded above by*

$$\mathcal{O}(T_{\text{avg}} + \log n \cdot w_{\text{max}})$$

where  $T_{\text{avg}} = \frac{1}{k} \sum_{i=1}^n w(v_i)$  and  $w_{\text{max}} = \max\{w(v_i)\}$ .

## Lemma (Mapping Lemma)

*Let an  $n$ -node,  $m$ -edge graph  $G$  be partitioned among the  $k$  machines as  $N = \{p_1, \dots, p_k\}$ . Then with probability at least  $1 - 1/n^\alpha$ , where  $\alpha > 1$  is an arbitrary fixed constant, the following bounds hold:*

- ① *The number of vertices mapped to any machine is  $\mathcal{O}(n/k)$ .*
- ② *The number of edges mapped to any machine is  $\mathcal{O}(m/k + \Delta \log n)$ .*
- ③ *The number of edges mapped to any link of the network is  $\mathcal{O}(m/k^2 + n/k)$ .*



# Table of Contents

- 1 Introduction
- 2 Parallel Flooding**
- 3 Filtering-Based MST
- 4 Deterministic Borøuvka-Style Algorithm
- 5 An Almost-Optimal Randomized Algorithm

- the algorithm can be described from the perspective of a node
  - each node chooses an ID uniformly at random in  $[1, n^4]$
  - each node floods its ID
  - upon receiving a message (with an ID), if the new ID is greater than the current ID, the node updates its ID and floods
- in order to simulate this in the  $k$ -Machine Model, each machine  $M$ 
  - maintains list of nodes on the machine in descending-order of ID
  - iterate through list and simulate each node individually
    - when  $u$  located on  $M$  sends a message to  $v$  on  $M'$ ,  $M$  sends the appropriate message to  $M'$
  - after some  $\mathcal{O}(n \log n/k + D)$  rounds, aggregate all IDs located on machine  $M$  and send them to machine  $M_1$
- Finally, machine  $M_1$  counts the number of distinct IDs (which is the number of connected components) and broadcasts it to all machines

## Theorem

*With high probability, the above algorithm correctly counts the number of connected components with*

$$T_\ell = \mathcal{O}((m/k + \Delta \log n) \log n + k)$$

$$T_c = \mathcal{O}(n \log n / k + D)$$

## Proof.

- with Chernoff Bounds, can show that a node  $v$  will update max-ID  $\mathcal{O}(\log n)$  times with probability  $1 - 1/n^a$
- $v$  will receive  $\mathcal{O}(\deg(v) \log n)$  higher IDs
- when max-ID is updated,  $v$  sends to  $\deg(v)$  nodes, totaling  $\mathcal{O}(\deg(v) \log n)$
- maximum runtime is therefore  $\mathcal{O}(\Delta \log n)$ .



Proof.

- apply node distribution lemma on the degree of nodes

$$\begin{aligned} T_\ell &= \mathcal{O}\left(\frac{1}{k} \left(\sum_{i=1}^n (d(v_i) \log n)\right) + \log n \cdot \Delta \log n\right) \\ &= \mathcal{O}\left(\left(\frac{m}{k} + \Delta \log n\right) \log n\right) \end{aligned}$$

- in the worst case, a machine sends/receives messages from all other machines, totaling

$$\mathcal{O}\left(\left(\frac{m}{k} + \Delta \log n\right) \log n + k\right)$$



## Lemma

*The communication complexity of the algorithm is  $\mathcal{O}(n \log n/k + D)$  with high probability.*

## Proof.

- by conversion theorem (see ), total number of broadcasts is  $\mathcal{O}(B \log n/(kW) + D)$  where  $W$  is the bandwidth
- taking  $W = \mathcal{O}(\log n)$
- by mapping lemma,  $\mathcal{O}(\log n)$  broadcasts for a particular node with probability  $1 - 1/n^2$
- by union bound,  $\mathcal{O}(\log n)$  broadcasts for all nodes with probability  $1 - 1/n$
- thus,  $\mathcal{O}(n \log n)$  broadcasts initiated with high probability



# Table of Contents

- 1 Introduction
- 2 Parallel Flooding
- 3 Filtering-Based MST**
- 4 Deterministic Borøuvka-Style Algorithm
- 5 An Almost-Optimal Randomized Algorithm

# Preliminaries

- uses three procedures:
- Kruskal's algorithm — outputs an MST in  $\mathcal{O}(m \log m)$  rounds
- maximal matching a clique of size  $n$ 
  - done by simply sorting IDs and matching node  $2i - 1$  to node  $2i$
- distributed routing in a clique
  - node  $u$  wants to send  $f$  messages to a different node  $v$
  - phase 1:
    - divide messages into groups of  $f/(n - 1)$
    - append destination id to messages
    - send one group per edge
  - phase 2:
    - each node forwards its message to the destination

# The Algorithm

- each machine  $M$  maintains
  - a graph  $G_M$ 
    - initially comprised of subgraph induced by edges in  $M$
  - ordered list  $H_M$  comprised of IDs of all machines
- each machine  $M$  executes at most  $2 \log k$  phases of:
  - if  $M$  received information about new edges in previous phase, it updates  $G_M$  accordingly, then runs Kruskal's to remove cycles
  - if  $|H_M| = 1$ , terminate. Use matching procedure on  $H_M$  to match with machine  $M'$ . If the ID of  $M$  is greater than that of  $M'$ , use routing procedure to send  $G_M$  to  $M'$
  - For any machine  $M''$  in  $H_M$ , if  $M''$  matched with a machine with a lower ID than  $M''$ , remove  $M''$  from  $H_M$ . If  $M$  is such a machine, terminate
- at the end of the algorithm, the machine with the lowest ID will have the entire MST



- Correctness is fairly straightforward
  - if  $e$  is a cut edge, then it will never be removed by Kruskal's
  - the matching algorithm guarantees that after  $2 \log k$  phases, the lowest ID machine will contain  $e$

# Table of Contents

- 1 Introduction
- 2 Parallel Flooding
- 3 Filtering-Based MST
- 4 Deterministic Borůvka-Style Algorithm**
- 5 An Almost-Optimal Randomized Algorithm

- we present two deterministic algorithms
- however, a quick review of Borøuvka's
- each machine stores a disjoint-set (union-find)  $\mathcal{M}$
- in each phase:
  - an MOE is found for each fragment
  - fragments are merged along MOEs

# Algorithm 1

- each phase consists of two steps
  - for any node  $v$  in  $M$ ,  $M$  iterates through the adjacency list of  $v$  and performs `FIND` until it finds an edge belonging to another fragment, after which it broadcasts this MOE
  - $M$  merges fragments: if  $(u, v)$  is an MOE,  $M$  updates  $\mathcal{M}$  with `UNION`( $u, v$ ).

## Algorithm 2 — The Improved Algorithm

- can improve  $T_\ell$
- rather than broadcasting MOEs, simulate unicast version (as in GHS)
- additionally, we filter to reduce the number of edges in each machine to  $\mathcal{O}(n)$ 
  - machines create MSFs of their local subgraphs using Kruskal's
  - “discard” edges that are not part of local MSF
  - the remaining (at most  $k(n-1)$  edges whp) are the only MST edges remaining
- algorithm then continues on remaining subgraphs

- for a fixed machine  $M$  and fragment  $f$
- set  $\text{LOE} = \infty$ 
  - iterate through edges in  $f$  — for edge  $(u, v)$  with  $u$  in  $M$ , send a message to the machine containing  $v$  with IDs of  $u$  and  $v$  and fragment ID of  $f$
  - for each message  $(u, v, f)$ , if the fragment ID of  $v$  is different than the received fragment ID, respond with original fragment ID and fragment ID of  $f$
  - update  $\text{LOE} = \text{MIN}(\text{LOE}, w)$  for each message received corresponding to  $f$ , where  $w$  is the weight of the outgoing edge
  - broadcast the LOE and fragment ID
- from the broadcast of the LOEs, each machine locally determines the global MOE for any fragment it contains
- the machine then merges fragments

# Merging

- Note that we cannot merge all fragments at once, as that takes time proportional to the length of the fragment chain
- use a technique similar to **controlled GHS**
  - create rooted tree  $F$  — each node is a fragment and there is an edge between nodes if they share an MOE
  - construct maximal matching (using e.g. Cole-Vishkin)
  - merge all matched edges and any edge where **exactly** one endpoint is matched





- we analyze the 2018 algorithm of Pandurangan, Robinson, and Scquizzato
- optimal up to polylog factors in terms of round complexity
- but  $\tilde{O}(n^2)$  local computation complexity
- the algorithm is similar to Borøuvka's algorithm, with the output being a **labeling** of the nodes, such that nodes in the same component have the same label
- component also have a **component proxy** that handles finding MOEs and merging for the component
- initially, each node has label equal to its ID (forming its own component) and is its own component proxy
- to load balance communication and computation, component proxies are chosen randomly among the machines
- given a component  $f$  and a machine  $M$ , the set of nodes belonging to  $f$  in  $M$  are called the **component part**
- relies on **linear graph sketches** to efficiently merge multiple components

- sketches are found by forming a vector of length  $\binom{n}{2}$  for each node  $v$  and an appropriate choice of an  $\mathcal{O}(\log^2 n) \times \binom{n}{2}$  matrix. This requires  $\tilde{\Omega}(n^2)$  local computation time.
- can be improved to  $\tilde{\mathcal{O}}((m+n)/k + \Delta + k)$  by using the 2011 sketches of Tardos, Saglam, and Jowhari
- instead of storing vectors of length  $\binom{n}{2}$ , a leader machine broadcasts an **odd hash function**

- each component uses this hash function to find an MOE as follows
  - each node in a component part evaluates  $\sum_{\text{incident edges } e} h(e) \bmod 2$
  - this sum is aggregated among the nodes in a component part by the corresponding machine
  - the sums of each part of the component are aggregated to the corresponding component proxy machine.
  - If the sum is 1, we conclude with some constant probability  $\varepsilon$  that an outgoing edge exists out of the component
    - By restricting the edges considered to be those within some range  $[i, j]$ , can (like binary search) determine an MOE with constant probability
    - repeating this process  $\Omega(\log n)$  times yields an MOE whp for a single component
    - by union bound, we have MOE for all components whp