Impromptu Updating of MST and ST in a Distributed Dynamic Graph

Valerie King, University of Victoria
Joint work with Ben Mountjoy, Mikkel Thorup and Shay Kutten
Network with \( n \) nodes, \( m \) edges. Each node has list of incident edges and edge weights. Nodes have distinct IDs.
A network maintains a subgraph if its edges are marked by their endpoints.
Communication: Each node may send messages of size $O(\log n)$ to all its neighbors in a single step.

Synchronous vs. Asynchronous
UPDATES:
Insert (\{A,D\}, edge\_weight)
Delete edge \{E,F\}
MST (resp. ST) Problem: Maintain a minimum spanning forest (resp., spanning forest) in a dynamic network
Main difficulty:
How to find a replacement edge when a tree edge is deleted
Our contribution:

A New Tool
for finding, impromptu, a replacement edge w.h.p.

for MST: an edge of min cost leaving a tree in a graph

for ST: any edge leaving a tree in a graph
Costs: MST / ST

Message complexity: $O(n \log n), O(n)$ (expected)

Preprocessing Time: NONE

Update Time: $O(\text{diam}(\text{tree}) \times \log n), O(\text{diam}(\text{tree}))$ expected.

Local memory needed between updates: NONE
previous distributed dynamic MST:

$O(n)$ messages--First dynamic updating
in $o(m)$ messages per update.

But local memory=
$O(n \times \text{degree of node} \times \log n)$
Stores the forest in each node;
Static MST/ST thought to require m messages!

Gallagher, Humblet and Spira (1983) 
$O(m + n \log n)$ messages for building one from scratch (asynchronously)

Our method yields $O(n \log^2 n)$ messages for constructing an MST in the synchronous model.

NOT KNOWN if m can be avoided for the asynchronous model.
Talk outline:

1. KEY IDEA
2. The Odd hash function
3. Updating MST
4. Static MST
5. Updating ST
6. OPEN problems for distributed and sequential dynamic graphs.
KEY IDEA:

- $C$ a maximally connected component of a graph. 
  → the sum of the degrees of the nodes in $C$ is \textbf{even}, since every edge incident to a node in $C$ contributes 2.

- If $C$ is not maximally connected, 
  → the sum of degrees of the nodes in $C$ of a random subset of edges is \underline{odd} with prob 1/2.
basic communication step: broadcast and return
How do we randomly sample and report results efficiently?
Odd hash function $F$

For any set $S$, we design hash

$$F: \{\text{weights}\} \rightarrow \{0, 1\}$$

s.t. there is a constant probability $(1/36)$ that an ODD number of its elements hash to 1, iff $S$ is non empty. Else it is 0.
Applying the Odd hash function $F$

Let $E'=$\{edges incident to nodes in tree $T_x$\}

$\text{XOR } F(e)$ (over all $e$ is incident to a node in $T_x$)

$= \text{XOR } F(e)$ (over all $e$ with one endpoint in $T_x$)

$= 1$ with prob. $1/36$ unless the cut is empty.
Using the ODD Hash function:
TEST if there is a Replacement edge

- When a tree edge \( \{X,Y\} \) is deleted, if \( X < Y \)
- \( X \) becomes leader, broadcasts Odd hash \( F \) to other nodes in tree \( T_x \).
- Each node applies \( F \) to their set of incident edges and computes the XOR;
- XOR is taken over all nodes in \( T_x \).
- Repeat in parallel \( O(\log n) \) times to get prob error \( 1/n^c \)
- Output 1 iff any one XOR =1
Find min wt replacement edge
(assuming distinct wts)

- Use binary search over the range of possible weights, testing w.h.p each time if there is a replacement edge in that wt range and narrowing the range.
- Return weight when only one is left.
Analysis

- $\lg (\text{Weight range})$ tests* cost of test

- \textbf{Cost of test} = initial cost of sending \log n hash functions, +
  + 1 broadcast and return for each phase of the binary search

- \textbf{Total} = $O(n \log^2 n)$ messages
Constructing the Odd hash function

- Let $U$ be the universe of elements.
- $S$ a subset of $U$.
- $F(x) \rightarrow \{0,1\}$
- We want:

$$\text{XOR}_{\{y \in S\}} F(y) = 1$$

iff $S$ is non empty
Odd hash function $F$

Obvious approach takes $O(\log n)$ hashes

$F$ has two parts,
- a 2-wise independent hash function $h: U \rightarrow U$
- $t$, a random element of $U$

DEF: $F(s)=1$ iff $h(s) < t$

Note: $F$ can be described in $O(\log n)$ bits.
Why F is an Odd hash function

- h hashes U → U
- Imagine 2|S| equal sized intervals.

T lands in some I, in top third or bottom third

Exactly one x in I, in middle third
CASE: F works if

Parity of elements hashed to intervals left of I is
- Odd and t is in bottom third or
- Even and t is in top third

T lands in some I, and either top third or bottom third of I

Exactly one x in I, x is in middle third
Static synchronous MST alg

While $I < \log n$

Repeat:

Each component finds min wt edge incident to it, sends message to other endpoint, and waits $n$ time steps. Then the found edges are inserted to form larger components.

Log $n$ phases, each takes $\log n$ broadcasts and returns, for a total of $O(n \log^2 n)$ expected message communication.
Find any edge in expected $O(1)$ broadcasts and returns

STEP 1: (IF step) Determine if there is a replacement edge w.h.p.
- Use deterministic amplification to send out $O(\log n)$ bits which can be used by individual nodes to deterministically generate $\log n$ Odd Hash functions s.t one is good w.h.p.
- Return $\log n$ outputs using ONE return
STEP 2: (find) If there is a replacement edge, find it

- Broadcast a single 2-wise independent hash function $h$
- For $i=0,\ldots, 2\lg n$, every node $x$ computes one word whose $i^{th}$ bit =
  \[
  \text{XOR}_{y \text{ incident to } x} h(y) \leq 2^i
  \]
- If XOR over tree $\neq 0$, $\min \leftarrow \text{first } i \neq 0$
- Test if there is exactly one edge with $h(y) \leq 2^\min$. If so, return it.
- Else Repeat find.
Open problem and discussion

- Can we avoid the $O(m)$ communication costs of Gallagher for the asynchronous static model?
- Why this method is more complicated for sequential dynamic graph problem
- How the sequential dynamic graph method is not fully understood
Open problems for Sequential dynamic ST

- How to apply it to MST
- How to bring it down to $O(\log^3 n)$?
- Can we remove the tiers?