Optimal Lower Bounds for Distributed and Streaming Spanning Forest Computation

Huacheng Yu
Oct 18, 2018
Harvard University

Joint work with Jelani Nelson
Warm-up

Consider the following dynamic problem:

- edges are inserted into an initially empty graph $G$ on $n$ vertices

Space complexity: $\Theta(n \log n)$ bits

- maintain list of edges in the spanning forest: $O(n \log n)$

- when the final graph is a tree itself, have to output the whole graph: $\Omega(n \log n)$

what if we allow edge deletions?
Consider the following dynamic problem:

- edges are inserted into an initially empty graph $G$ on $n$ vertices
- must output a spanning forest when queried

Goal: minimize space

- Space complexity: $\Theta(n \log n)$ bits
- maintain list of edges in the spanning forest: $O(n \log n)$
- when the final graph is a tree itself, have to output the whole graph: $\Omega(n \log n)$

what if we allow edge deletions?
Consider the following dynamic problem:

- edges are inserted into an initially empty graph $G$ on $n$ vertices
- must output a spanning forest when queried

Goal: minimize space
Warm-up

Consider the following dynamic problem:

- edges are inserted into an initially empty graph $G$ on $n$ vertices
- must output a spanning forest when queried

Goal: minimize space

Space complexity: $\Theta(n \log n)$ bits

- maintain list of edges in the spanning forest: $O(n \log n)$
- when the final graph is a tree itself, have to output the whole graph: $\Omega(n \log n)$
Consider the following dynamic problem:

- edges are inserted into an initially empty graph $G$ on $n$ vertices
- must output a spanning forest when queried

Goal: minimize space

Space complexity: $\Theta(n \log n)$ bits

- maintain list of edges in the spanning forest: $O(n \log n)$
- when the final graph is a tree itself, have to output the whole graph: $\Omega(n \log n)$

what if we allow edge deletions?
Fully dynamic spanning forest

Maintain a dynamic graph on $n$ vertices, supporting

- edge insertions,
- edge deletions, and
- spanning forest queries

Goal: minimize space

Theorem (Ahn, Guha, McGregor’12)

... solvable using $O(n \log^3 n)$ bits of space with error probability $1/poly(n)$. 
Fully dynamic spanning forest

Maintain a dynamic graph on $n$ vertices, supporting

- edge insertions,
- edge deletions, and
- spanning forest queries

Goal: minimize space

Theorem (Ahn, Guha, McGregor’12)

... solvable using $O(n \log^3 n)$ bits of space with error probability $1/poly(n)$.

only two more log factors!
Fully dynamic spanning forest

Maintain a dynamic graph on $n$ vertices, supporting

- edge insertions,
- edge deletions, and
- spanning forest queries

Goal: minimize space

Theorem (Ahn, Guha, McGregor’12)

... solvable using $O(n \log(n/\delta) \log^2 n)$ bits of space with error probability $\delta$.

only two more log factors!
Fully dynamic spanning forest

Maintain a dynamic graph on $n$ vertices, supporting

- edge insertions,
- edge deletions, and
- spanning forest queries

Goal: minimize space

Theorem (Ahn, Guha, McGregor’12)

... solvable using $O(n \log(n/\delta) \log^2 n)$ bits of space with error probability $\delta$.

only two more log factors! why two more?
Main result I

**Theorem (This paper)**

Any data structure for *fully dynamic spanning forest* with error probability $\delta$ must use $\Omega(n \log(n/\delta) \log^2 n)$ bits of memory, for any $2^{-n^{0.99}} < \delta < 0.99$. 

$\delta$ is a constant $\Rightarrow \Omega(n \log^3 n)$ bits of space: need exactly two more log factors!
Main result I

**Theorem (This paper)**

Any data structure for *fully dynamic spanning forest* with error probability $\delta$ must use $\Omega\left(n \log(n/\delta) \log^2 n\right)$ bits of memory, for any $2^{-n^{0.99}} < \delta < 0.99$.

$\delta$ is a constant $\implies \Omega(n \log^3 n)$ bits of space:

need *exactly two more* log factors!
Simultaneous communication

The [Ahn, Guha, McGregor’12] solution also solves the following $n$-player communication problem
Simultaneous communication

The [Ahn, Guha, McGregor’12] solution also solves the following $n$-player communication problem

A (fixed) graph on $n$ vertices is given to $n$ players w. shared randomness:

• each player only sees one vertex and its neighborhood
Simultaneous communication

The [Ahn, Guha, McGregor’12] solution also solves the following $n$-player communication problem

A (fixed) graph on $n$ vertices is given to $n$ players w. shared randomness:

- each player only sees one vertex and its neighborhood
- each player sends a message to a referee
Simultaneous communication

The [Ahn, Guha, McGregor’12] solution also solves the following $n$-player communication problem

A (fixed) graph on $n$ vertices is given to $n$ players w. shared randomness:

- each player only sees one vertex and its neighborhood
- each player sends a message to a referee
- referee outputs a spanning forest w.p. $1 - \delta$
Simultaneous communication

The [Ahn, Guha, McGregor’12] solution also solves the following $n$-player communication problem

A (fixed) graph on $n$ vertices is given to $n$ players w. shared randomness:

- each player only sees one vertex and its neighborhood
- each player sends a message to a referee
- referee outputs a spanning forest w.p. $1 - \delta$

Goal: minimize communication
Simultaneous communication

The [Ahn, Guha, McGregor’12] solution also solves the following \( n \)-player communication problem

A (fixed) graph on \( n \) vertices is given to \( n \) players w. shared randomness:

- each player only sees one vertex and its neighborhood
- each player sends a message to a referee
- referee outputs a spanning forest w.p. \( 1 - \delta \)

Goal: minimize communication

(compute a global function given small “sketches” of “local information”)

A graph on $n$ vertices is given to $n$ players w. shared randomness:

- each player only sees one vertex and its neighborhood
- each player sends a message to a referee
- referee outputs a spanning forest w.p. $1 - \delta$

Goal: minimize communication
A graph on \( n \) vertices is given to \( n \) players w. shared randomness:

- each player only sees one vertex and its neighborhood
- each player sends a message to a referee
- referee outputs a spanning forest w.p. \( 1 - \delta \)

Goal: minimize communication

Theorem (AGM’12)

... solvable using (worst-case) \( O(\log(n/\delta) \log^2 n) \) bits of communication per player with error probability \( \delta \).
AGM sketch for simultaneous communication

A graph on \( n \) vertices is given to \( n \) players w. shared randomness:

- each player only sees one vertex and its neighborhood
- each player sends a message to a referee
- referee outputs a spanning forest w.p. \( 1 - \delta \)

Goal: minimize communication

Theorem (AGM’12)

... solvable using (worst-case) \( O(\log(n/\delta) \log^2 n) \) bits of communication per player with error probability \( \delta \).

Trivial: \( \Omega(\log n) \) since the referee has to learn \( \Omega(n \log n) \) bits
Main result II

**Theorem (This paper)**

Any simultaneous communication protocol for spanning forest with error probability 0.99 must use $\Omega(\log^3 n)$ bits of communication on average.
Main result II

**Theorem (This paper)**

Any simultaneous communication protocol for spanning forest with error probability 0.99 must use $\Omega(\log^3 n)$ bits of communication on average.

*exactly two more log factors needed* than the trivial information theoretical lower bound
Main result II

Theorem (This paper)

Any simultaneous communication protocol for spanning forest with error probability 0.99 must use $\Omega(\log^3 n)$ bits of communication on average.

Exactly two more log factors needed than the trivial information theoretical lower bound

Open: higher lower bounds when error probability $\delta$ is lower?
[AGM’12] designed a (randomized) linear sketch:

\[ S : \mathbb{N}^{n^2} \rightarrow \mathbb{N}^{O(n \log^2 n)} \]

such that
Graph sketching for spanning forest

[AGM’12] designed a (randomized) linear sketch:

\[ S : \mathbb{N}^{n^2} \rightarrow \mathbb{N}^{O(n \log^2 n)} \]

such that

- \( S \) is a linear mapping with poly-bounded coefficients
[AGM’12] designed a (randomized) linear sketch:

\[ S : \mathbb{N}^{n^2} \rightarrow \mathbb{N}^{O(n \log^2 n)} \]

such that

- \( S \) is a linear mapping with poly-bounded coefficients
- \( S(G) \) is a concatenation of \( S_1(G), S_2(G), \ldots, S_n(G) \), each \( S_i(G) \) has \( O(\log^2 n) \) dimensions,
  and it is computed from the neighborhood of vertex \( i \)
[AGM’12] designed a (randomized) linear sketch:

\[ S : \mathbb{N}^{n^2} \rightarrow \mathbb{N}^{O(n \log^2 n)} \]

such that

- \( S \) is a linear mapping with poly-bounded coefficients
- \( S(G) \) is a concatenation of \( S_1(G), S_2(G), \ldots, S_n(G) \), each \( S_i(G) \) has \( O(\log^2 n) \) dimensions,
  and it is computed from the neighborhood of vertex \( i \)
- \( S(G) \) determines a spanning forest with probability \( 1 - 1/n^c \)
Store $S(G)$ in memory:

- update: $S(G \pm (u, v)) = S(G) \pm S((u, v))$
- at end of stream: $S(G)$ determines a spanning forest w.h.p.

Use $O(n \log^3 n)$ bits of space
Communication protocol

Given graph $G$:

- Player $i$ computes $S_i(G)$, and sends it to referee
- referee concatenates all $S_i(G)$, obtains $S(G)$
- referee outputs a spanning forest w.h.p.

Use $O(\log^3 n)$ bits of communication per player
Simultaneous communication complexity of spanning forest
Recall...

An $n$-vertex graph is given to $n$ players with shared randomness:

- each player only sees one vertex and its neighborhood
- each player sends a message to a referee
- referee outputs a spanning forest w.p. $1 - \delta$

Goal: prove an average player must send $\Omega(\log^3 n)$ bits for constant $\delta$
An $n$-vertex graph is given to $n$ players with shared randomness:

- each player only sees one vertex and its neighborhood
- each player sends a message to a referee
- referee outputs a spanning forest w.p. $1 - \delta$

Goal: prove some player must send $\Omega(\log^3 n)$ bits for $\delta = 1/n^c$
An $n$-vertex graph is given to $n$ players with shared randomness:

- each player only sees one vertex and its neighborhood
- each player sends a message to a referee
- referee outputs a spanning forest w.p. $1 - \delta$

Goal: prove some player must send $\Omega(\log^3 n)$ bits for $\delta = 1/n^c$

Starting point: Universal Relation $UR^\supset$...
Universal Relation $\mathsf{UR}^D$

Alice: $S \subseteq [n]$

Bob: $T \subset S$

$M$

shared random bits...

output any $x \in S \setminus T$

Theorem (KNPWWY'17)

For failure probability $\delta > 2^{-n^{0.99}}$, the optimal length of $M$ is $\Theta(\log(1/\delta) \log 2^n)$.

In particular, if $1/n$ failure probability, optimal length is $\Theta(\log 3^n)$. 

12
Universal Relation $\mathbb{UR}^\subset$

**Theorem (KNPWWY’17)**

*For failure probability $\delta > 2^{-n^{0.99}}$, the optimal length of $M$ is $\Theta(\log(1/\delta) \log^2 n)$.***
Universal Relation $\mathcal{UR}^\supset$

**Theorem (KNPWWY'17)**

For failure probability $\delta > 2^{-n^{0.99}}$, the optimal length of $M$ is $\Theta(\log(1/\delta) \log^2 n)$.

In particular, $1/n^c$ failure probability, optimal length is $\Theta(\log^3 n)$. 
Connection to $\mathbb{UR}^3$.

The referee $M$ is only a neighbor of $v$. $S$ is the set of neighbors of $v$ and $T$ is the set of vertices that $v$ is only a neighbor of. The referee has to find some element in $S \setminus T$.

Why not already an $\Omega(\log^3 n)$ LB?

$u_1$ may also reveal $(v, u_1)$...
Connection to $\mathbb{UR}^{\Box}$

$v \succ u_1, u_2, \ldots, u_k$

Referee has to find some element in $S \setminus T$.

Why not already an $\Omega(\log^3 n)$ LB?

$M_v$ may also reveal $(v, u_1)$...
Connection to $\text{UR}^{3}$

$v \subseteq u_1, u_2, \ldots, u_k$

Referee has to find some element in $S \setminus T$.

Why not already an $\Omega(\log^3 n)$ LB?

$M_v$ may also reveal $(v, u_1)$...
Connection to $\mathbb{UR}^3$

$v$ is only neighbor

$u_1 \rightarrow u_2 \rightarrow \ldots \rightarrow u_k \rightarrow v$

$M_v$ to referee

$v$ is only neighbor

$\Omega(\log^3 n)$ LB
Connection to $\text{UR}^3$

$S$: neighbors of $v$

$T$: vertices that $v$ is only neighbor

$v$ is only neighbor

Referee has to find some element in $S \setminus T$.

Why not already an $\Omega(\log^3 n)$ LB?

$M_v$ may also reveal ($v$, $u_1$),...
Connection to $\mathbb{UR}^k$

$v$ is only neighbor

$S$: neighbors of $v$

$T$: vertices that $v$ is only neighbor

Referee has to find some element in $S \setminus T$. Why not already an $\Omega(\log^3 n)$ LB?

$M_v$ may also reveal $(v, u_1, \ldots, u_k)\ldots$
Referee has to find some element in $S \setminus T$.

Why not already an $\Omega(\log^3 n)$ LB?
Connection to $\text{UR}^3$

Referee has to find some element in $S \setminus T$.

Why not already an $\Omega(\log^3 n)$ LB? $M_{u_1}$ may also reveal $(v, u_1)$...
Hard instances

\[ n - \epsilon |V_r| = n \epsilon |V_i| v_i \]

For vertex \( v_i \), its neighbors encode set \( S_i \), its neighbors on the left encode set \( T_i \).

Spanning forest contains an edge between \( v_i \) and \( V_r \).
Hard instances

For vertex $v_i$, its neighbors encode set $S_i$, its neighbors on the left encode set $T_i$. Spanning forest contains an edge between $v_i$ and $V_{r \epsilon}$. 

$n^{1-\epsilon}$
Hard instances

For vertex $v_i$, its neighbors encode set $S_i$, its neighbors on the left encode set $T_i$. Spanning forest contains an edge between $v_i$ and $V_r$. 

$n^{1-\epsilon}$

$|V_r| = n^\epsilon$
Hard instances

$|V_r| = n^\epsilon$

$V_i$

$n^\epsilon$
Hard instances

\[ n^{1 - \varepsilon} | V_r | = n^\varepsilon \]

vertices randomly permuted
Hard instances

Vertices randomly permuted

For vertex $v_i$, its neighbors encode set $S_i$
Hard instances

vertices randomly permuted

For vertex $v_i$, its neighbors encode set $S_i$, its neighbors on the left encode set $T_i$. 
Hard instances

For vertex $v_i$, its neighbors encode set $S_i$, its neighbors on the left encode set $T_i$.

Spanning forest contains an edge between $v_i$ and $V_r$. 

vertices randomly permuted
Generating hard instances:

1. Fix \{v_i\} arbitrarily, randomly partition the rest into \{V_i\}, \ V_r;
Generating hard instances:

1. Fix \( \{v_i\} \) arbitrarily, randomly partition the rest into \( \{V_i\}, \ V_r; \)
2. For each \( v_i \), generate \( S_i, T_i \) from hard distribution for \( UR^\ominus; \)

\[ \text{Diagram showing connections between } V_i, T_i, S_i, \text{ and } V_r. \]
Generating hard instances:

1. Fix \{v_i\} arbitrarily, randomly partition the rest into \{V_i\}, \ V_r;
2. For each \ v_i, generate \ S_i, T_i \ from hard distribution for UR\supseteq; 
3. Connect each \ v_i \ to |T_i| \ random vertices in \ V_i; 
4. Connect each \ v_i \ to |S_i \setminus T_i| \ random vertices in \ V_r.
Reduction from $\mathsf{UR}^\triangleright$

Make a reduction from $\mathsf{UR}^\triangleright$, main idea to solve $\mathsf{UR}^\triangleright$:

embed input $(S, T)$ into one of $(S_i, T_i)$,

then simulate the spanning forest protocol.
Reduction from $\text{UR}^<$

Make a reduction from $\text{UR}^<$, main idea to solve $\text{UR}^<$:

embed input $(S, T)$ into one of $(S_i, T_i)$,

then simulate the spanning forest protocol.

Goals:

1. Generate a graph $G$ that “looks like” a hard instance
Reduction from $\text{UR}^\supset$

Make a reduction from $\text{UR}^\supset$, main idea to solve $\text{UR}^\supset$:

embed input $(S, T)$ into one of $(S_i, T_i)$,

then simulate the spanning forest protocol.

Goals:

1. Generate a graph $G$ that “looks like” a hard instance
2. Spanning forest tells us an element in $S \setminus T$
Reduction from $UR^\supset$

Make a reduction from $UR^\supset$, main idea to solve $UR^\supset$:

embed input $(S, T)$ into one of $(S_i, T_i)$,

then simulate the spanning forest protocol.

Goals:

1. Generate a graph $G$ that “looks like” a hard instance
2. Spanning forest tells us an element in $S \setminus T$
3. Low communication cost and preserve success probability
Given \((S, T)\) over universe \([n^\epsilon]\), generate a random graph \(G\):

1. Sample a random \(v_i\), a random injection \(f : [n^\epsilon] \rightarrow \mathcal{V} \setminus \{v_i\}_i\)
Solving \( \mathbf{UR}^\square \)

Given \((S, T)\) over universe \([n^\epsilon]\), generate a random graph \(G\):

1. Sample a random \(v_i\), a random injection \(f: [n^\epsilon] \rightarrow V \setminus \{v_i\}_i\)
2. Connect \(v_i\) to \(f(S)\)
Given \((S, T)\) over universe \([n^e]\), generate a random graph \(G\):

1. Sample a random \(v_i\), a random injection \(f : [n^e] \rightarrow V \setminus \{v_i\}\)
2. Connect \(v_i\) to \(f(S)\)
3. \(V_i := f(T) \cup (n^e - |T| \text{ other vertices})\)
Given \((S, T)\) over universe \([n^e]\), generate a random graph \(G\):

1. Sample a random \(v_i\), a random injection \(f : [n^e] \rightarrow V \setminus \{v_i\}_i\)
2. Connect \(v_i\) to \(f(S)\)
3. \(V_i := f(T) \cup (n^e - |T| \text{ other vertices})\)
4. \(V_r := f([n^e] \setminus T) \cup (|T| \text{ other vertices})\)
Given \((S, T)\) over universe \([n^e]\), generate a random graph \(G\):

1. Sample a random \(v_i\), a random injection \(f : [n^e] \rightarrow V \setminus \{v_i\}_i\)
2. Connect \(v_i\) to \(f(S)\)
3. \(V_i := f(T) \cup (n^e - |T| \text{ other vertices})\)
4. \(V_r := f([n^e] \setminus T) \cup (|T| \text{ other vertices})\)
5. Randomly partition other vertices into \(V_1, \ldots, V_{i-1}, V_{i+1}, \ldots\), sample the neighborhoods of \(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots\)
Given \((S, T)\) over universe \([n^e]\), generate a random graph \(G\):

1. Sample a random \(v_i\), a random injection \(f : [n^e] \rightarrow V \setminus \{v_i\}\)
2. Connect \(v_i\) to \(f(S)\)
3. \(V_i := f(T) \cup (n^e - |T| \text{ other vertices})\)
4. \(V_r := f([n^e] \setminus T) \cup (|T| \text{ other vertices})\)
5. Randomly partition other vertices into \(V_1, \ldots, V_{i-1}, V_{i+1}, \ldots\), sample the neighborhoods of \(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots\)

Distribution of \(G\) is the hard distribution.
Given \((S, T)\) over universe \([n^\epsilon]\), generate a random graph \(G\):

1. Sample a random \(v_i\), a random injection \(f : [n^\epsilon] \rightarrow V \setminus \{v_i\}_i\)
2. Connect \(v_i\) to \(f(S)\)
3. \(V_i := f(T) \cup (n^\epsilon - |T|\) other vertices\)
4. \(V_r := f([n^\epsilon] \setminus T) \cup (|T|\) other vertices\)
5. Randomly partition other vertices into \(V_1, \ldots, V_{i-1}, V_{i+1}, \ldots\)
   sample the neighborhoods of \(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots\)

Distribution of \(G\) is the hard distribution.

Let \(u\) be one \(v_i\)'s neighbor in \(V_r\), then \(f^{-1}(u) \in S \setminus T\).
Given \((S, T)\) over universe \([n^\epsilon]\)

A: send \(M_{v_i}\) based on \(f(S)\)

B: analyze the distribution of \(G\) conditioned on \(f, T, M_{v_i}\)

B: find \(u \in V_r\) that is a neighbor of \(v_i\) with the highest prob., output \(f^{-1}(u)\)

\[ V_r = f([n^\epsilon] \setminus T) \cup (|T| \text{ other vertices}) \]
Analyzing the protocol

The protocol for $\text{UR}^\supset$ has

- communication cost $|M_{v_i}|$, and
- failure probability $\leq \delta + 1/n^{0.1}$.

By [KNPWY'17], $|M_{v_i}| \geq \Omega(\log(1/\delta) \log^2 n)$

($\Omega(\log^3 n)$ lower bound when $\delta = 1/n^c$)
Open question

Lower bounds for simultaneous communication when error probability is small? \( \Omega(\log(n/\delta) \log^2 n) \)?
Open question

Lower bounds for simultaneous communication when error probability is small? $\Omega(\log(n/\delta) \log^2 n)$?

Proving the same lower bounds for maintaining connected components? and for connectivity: “if the whole graph is connected”?
Open question

Lower bounds for simultaneous communication when error probability is small? $\Omega(\log(n/\delta) \log^2 n)$?

Proving the same lower bounds for maintaining connected components? and for connectivity: “if the whole graph is connected”?

Thank you!