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

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

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**

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: $\Omega(n \log n)$

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: $\Omega(n \log n)$

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 $1/\mathrm{poly}(n)$.

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 $1/\mathrm{poly}(n)$.

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 δ.

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 δ.

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 δ must use $\Omega(n \log(n/\delta) \log^2 n)$ bits of memory, for any $2^{-n^{0.99}} < \delta < 0.99$.

Main result I

Theorem (This paper)

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

δ 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")

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**

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 δ.

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 δ.

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 δ is lower?

Graph sketching for spanning forest

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

$$S : \mathbb{N}^{n^2} \to \mathbb{N}^{O(n \log^2 n)}$$

such that

Graph sketching for spanning forest

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

$$S : \mathbb{N}^{n^2} \to \mathbb{N}^{O(n \log^2 n)}$$

such that

- S is a linear mapping with poly-bounded coefficients

Graph sketching for spanning forest

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

$$S : \mathbb{N}^{n^2} \to \mathbb{N}^{O(n \log^2 n)}$$

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

Graph sketching for spanning forest

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

$$S : \mathbb{N}^{n^2} \to \mathbb{N}^{O(n \log^2 n)}$$

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 $1 - 1/n^c$

Streaming algorithm

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 δ

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

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

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

Universal Relation $\mathrm{UR}^{\supset}$

shared random bits...

Alice: $S \subseteq [n]$ $\xrightarrow{\mathbf{M}}$ Bob: $T \subset S$

output any $x \in S \setminus T$

Universal Relation $\mathrm{UR}^{\supset}$

shared random bits...

Alice: $S \subseteq [n]$ $\xrightarrow{\mathbf{M}}$ Bob: $T \subset S$

output any $x \in S \setminus T$

Theorem (KNPWWY'17)

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

Universal Relation $\mathrm{UR}^{\supset}$

shared random bits...

$\text{Alice: } S \subseteq [n] \xrightarrow{\mathbf{M}} \text{Bob: } T \subset S$

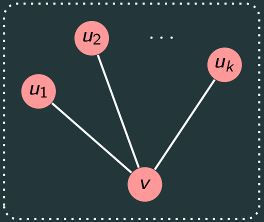
output any $x \in S \setminus T$

Theorem (KNPWWY'17)

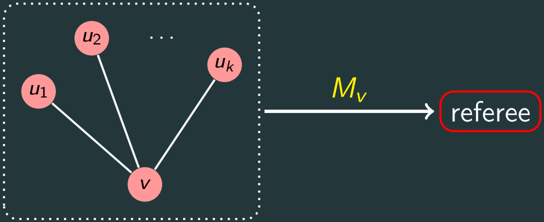
For failure probability $\delta > 2^{-n^{0.99}}$, the optimal length of $\mathbf{M}$ is $\Theta(\log(1/\delta)\log^2 n)$.

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

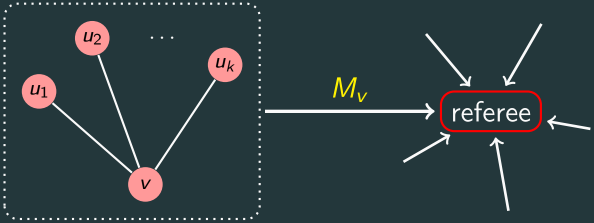
Connection to $\mathsf{UR}^{\supset}$



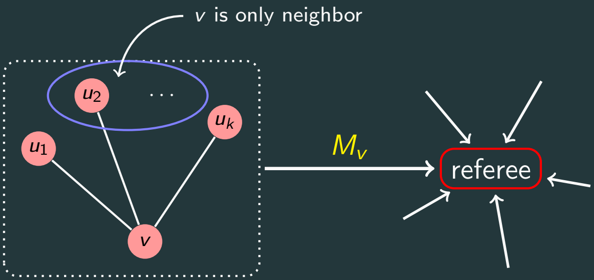
Connection to $\mathrm{UR}^{\supset}$



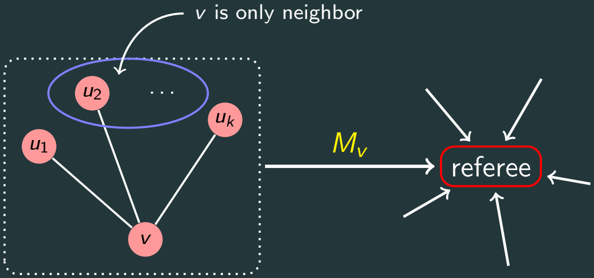
Connection to $\mathrm{UR}^{\supset}$



Connection to $\mathsf{UR}^{\supset}$

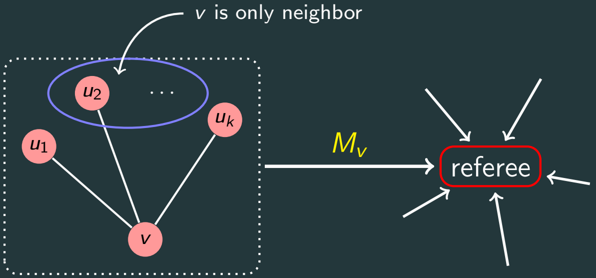


Connection to $\mathsf{UR}^{\supset}$



S: neighbors of v T: vertices that v is only neighbor

Connection to $\mathrm{UR}^{\supset}$

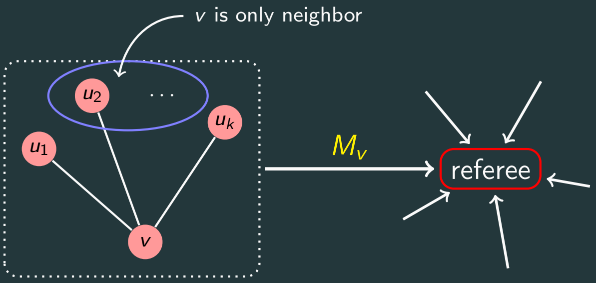


S: neighbors of v

T: vertices that v is only neighbor

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

Connection to $\mathrm{UR}^{\supset}$



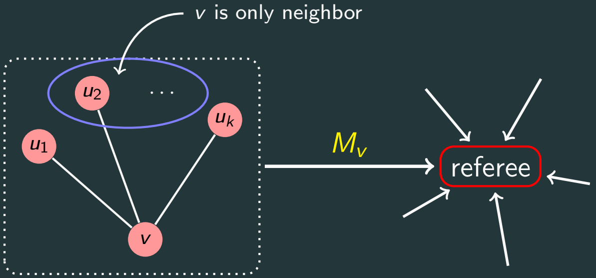
S: neighbors of v

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?

Connection to UR$^{\supset}$

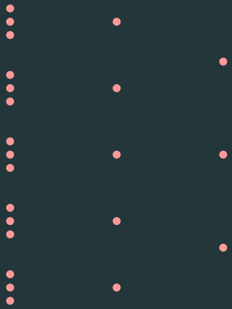


S: neighbors of v $\qquad$ T: vertices that v is only neighbor

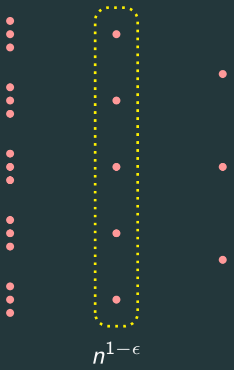
Referee has to find some element in $S \setminus T$.

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

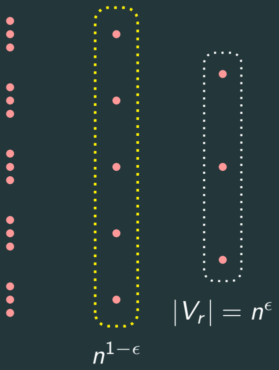
Hard instances



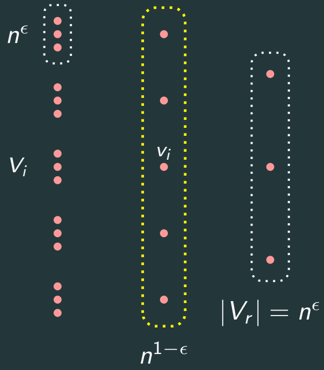
Hard instances



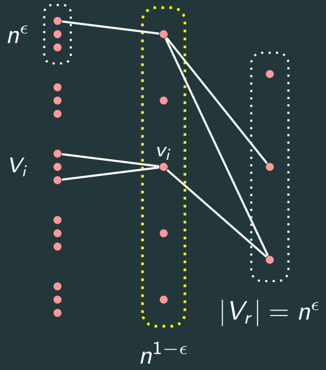
Hard instances



Hard instances

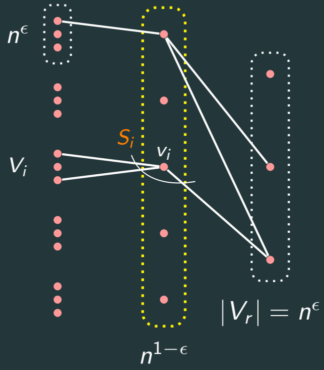


Hard instances



vertices randomly permuted

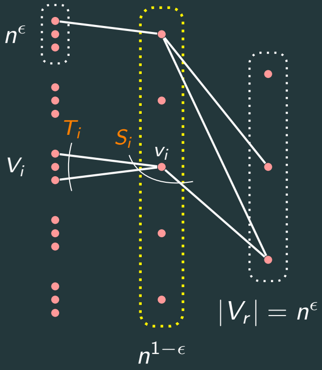
Hard instances



vertices randomly permuted

For vertex v_i, its neighbors encode set S_i

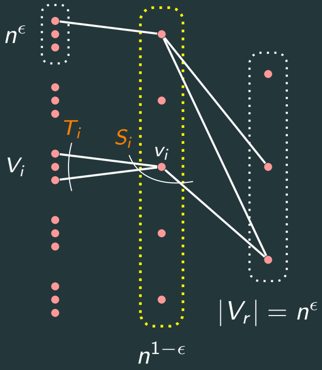
Hard instances



vertices randomly permuted

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

Hard instances



vertices randomly permuted

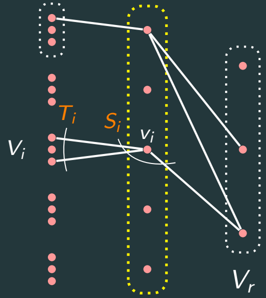
For vertex v_i, its neighbors encode set S_i, its neighbors on the left encode set T_i.

Spanning forest contains an edge between v_i and V_r.

Hard distribution

Generating hard instances:

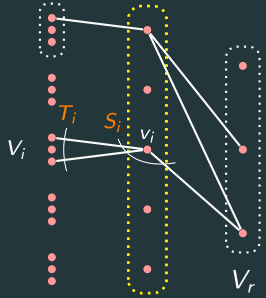
1. Fix $\{v_i\}$ arbitrarily, randomly partition the rest into $\{V_i\}$, V_r;



Hard distribution

Generating hard instances:

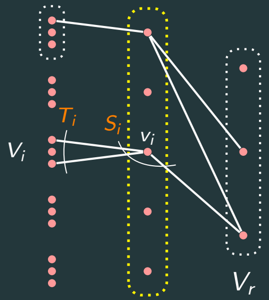
1. Fix $\{v_i\}$ arbitrarily, randomly partition the rest into $\{V_i\}$, V_r;
2. For each v_i, generate S_i, T_i from hard distribution for $\mathrm{UR}^{\supset}$;



Hard distribution

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 $\mathrm{UR}^{\supset}$;
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 $\mathrm{UR}^{\supset}$

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

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

then simulate the spanning forest protocol.

Reduction from $\mathrm{UR}^{\supset}$

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

$$\text{embed input } (S, T) \text{ into one of } (S_i, T_i),$$

then simulate the spanning forest protocol.

Goals:

1. Generate a graph G that "looks like" a hard instance

Reduction from $\mathrm{UR}^{\supset}$

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

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 $\mathsf{UR}^{\supset}$

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

$$\text{embed input } (S, T) \text{ into one of } (S_i, T_i),$$

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

Solving $\mathrm{UR}^{\supset}$

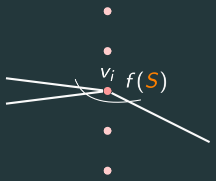
Given (S, T) over universe $[n^{\epsilon}]$, generate a random graph G:

1. Sample a random v_i, a random injection $f : [n^{\epsilon}] \to V \setminus \{v_i\}_i$

Solving UR$^{\supset}$

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

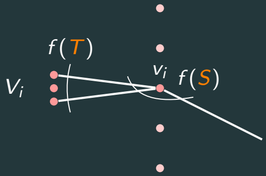
1. Sample a random v_i, a random injection $f : [n^{\epsilon}] \to V \setminus \{v_i\}_i$
2. Connect v_i to $f(S)$



Solving $\mathsf{UR}^{\supset}$

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

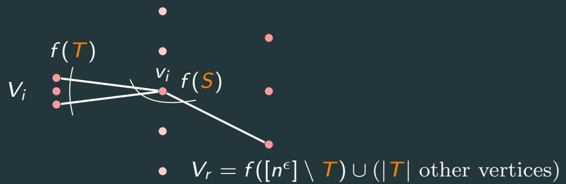
1. Sample a random v_i, a random injection $f : [n^\epsilon] \to V \setminus \{v_i\}_i$
2. Connect v_i to $f(S)$
3. $V_i := f(T) \cup (n^\epsilon - |T|\ \text{other vertices})$



Solving $\mathsf{UR}^{\supset}$

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

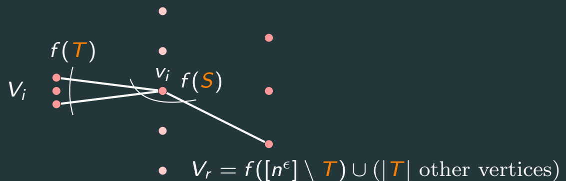
1. Sample a random v_i, a random injection $f : [n^\epsilon] \to V \setminus \{v_i\}_i$
2. Connect v_i to $f(S)$
3. $V_i := f(T) \cup (n^\epsilon - |T| \text{ other vertices})$
4. $V_r := f([n^\epsilon] \setminus T) \cup (|T| \text{ other vertices})$



Solving UR$^{\supset}$

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

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



Solving $\mathrm{UR}^{\supset}$

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

1. Sample a random v_i, a random injection $f : [n^\epsilon] \to V \setminus \{v_i\}_i$
2. Connect v_i to $f(S)$
3. $V_i := f(T) \cup (n^\epsilon - |T| \text{ other vertices})$
4. $V_r := f([n^\epsilon] \setminus T) \cup (|T| \text{ other vertices})$
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.

Solving $\mathrm{UR}^{\supset}$

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

1. Sample a random v_i, a random injection $f : [n^\epsilon] \to V \setminus \{v_i\}_i$
2. Connect v_i to $f(S)$
3. $V_i := f(T) \cup (n^\epsilon - |T| \text{ other vertices})$
4. $V_r := f([n^\epsilon] \setminus T) \cup (|T| \text{ other vertices})$
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.

Let u be one v_i's neighbor in V_r, then $f^{-1}(u) \in S \setminus T$.

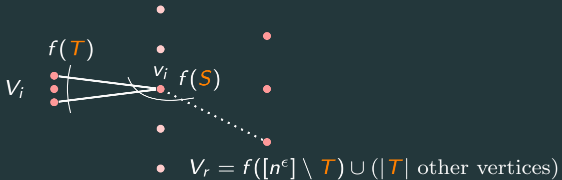
$\mathrm{UR}^{\supset}$ **protocol**

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 $f^{-1}(u)$



Analyzing the protocol

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

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

By [KNPWWY'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!