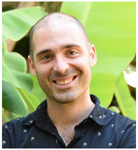
Universally-Optimal Distributed Algorithms for Known Topologies

Speaker: Goran Zuzic

STOC 2021

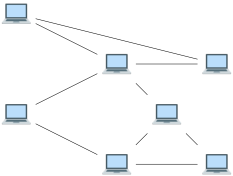

Bernhard Haeupler
CMU / ETH Zürich


David Wajc
Stanford


Goran Zuzic
ETH Zürich

Introduction

Setting: we are given some specific distributed network G.

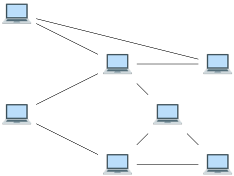


Examples:

- Computers in a network.
- Processors in a supercomputer.
- Sensors in a field.

Introduction

Setting: we are given some specific distributed network G.



Examples:

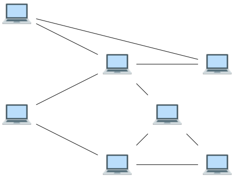
- Computers in a network.
- Processors in a supercomputer.
- Sensors in a field.

Problem: solve an optimization problem on the network graph.

- MST (minimum spanning tree),
- SSSP (single-source shortest path),
- Mincut, etc.

Introduction

Setting: we are given some specific distributed network G.



Examples:

- Computers in a network.
- Processors in a supercomputer.
- Sensors in a field.

Problem: solve an optimization problem on the network graph.

- MST (minimum spanning tree),
- SSSP (single-source shortest path),
- Mincut, etc.

Goal

Design a distributed MST protocol that is as fast as possible on G.

Why is distributed optimization important?

- The world is becoming more-and-more decentralized.

Why is distributed optimization important?

- The world is becoming more-and-more decentralized.
- Theoretical perspective: understanding fundamental barriers in distributed computing will help us design better algorithms.

Why is distributed optimization important?

- The world is becoming more-and-more decentralized.
- Theoretical perspective: understanding fundamental barriers in distributed computing will help us design better algorithms.
- Why MST? It is the most well-known and studied problem in the field. Introduced by [Gallagher, Humblet, Spira, 1983].

Why is distributed optimization important?

- The world is becoming more-and-more decentralized.
- Theoretical perspective: understanding fundamental barriers in distributed computing will help us design better algorithms.
- Why MST? It is the most well-known and studied problem in the field. Introduced by [Gallagher, Humblet, Spira, 1983].
- In spite of that, many important open questions remain.

Why is distributed optimization important?

- The world is becoming more-and-more decentralized.
- Theoretical perspective: understanding fundamental barriers in distributed computing will help us design better algorithms.
- Why MST? It is the most well-known and studied problem in the field. Introduced by [Gallagher, Humblet, Spira, 1983].
- In spite of that, many important open questions remain.
- Practical perspective: spanning tree protocol.

Why is distributed optimization important?

- The world is becoming more-and-more decentralized.
- Theoretical perspective: understanding fundamental barriers in distributed computing will help us design better algorithms.
- Why MST? It is the most well-known and studied problem in the field. Introduced by [Gallagher, Humblet, Spira, 1983].
- In spite of that, many important open questions remain.
- Practical perspective: spanning tree protocol.

Why is distributed optimization important?

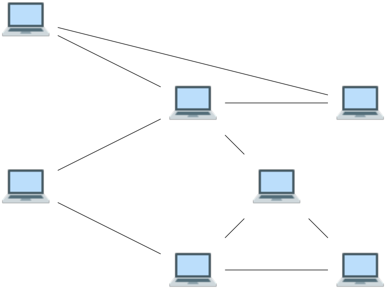
- The world is becoming more-and-more decentralized.
- Theoretical perspective: understanding fundamental barriers in distributed computing will help us design better algorithms.
- Why MST? It is the most well-known and studied problem in the field. Introduced by [Gallagher, Humblet, Spira, 1983].
- In spite of that, many important open questions remain.
- Practical perspective: spanning tree protocol.

Most prior work focuses only on pathological worst-case graphs G.

Our question: what is the optimal running time for non-worst-case networks G.

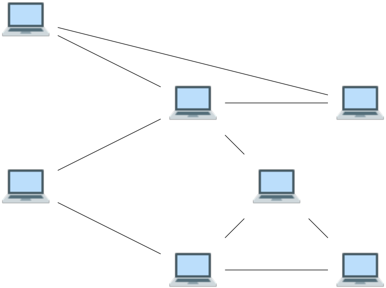
Model

CONGEST model



Your favorite large network G.

Model

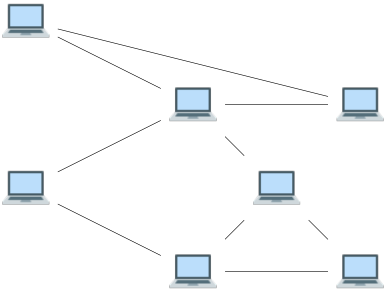


Your favorite large network G.

CONGEST model

- Network topology is an undirected graph.

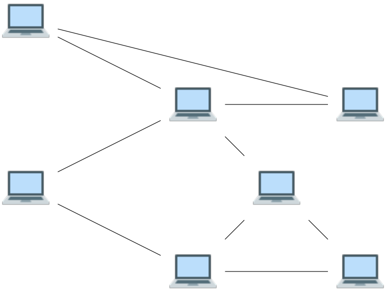
Model



Your favorite large network G.

CONGEST model

- Network topology is an undirected graph.
- Communication in synchronous rounds.

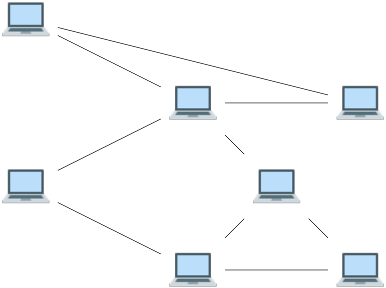


Your favorite large network G.

CONGEST model

- Network topology is an undirected graph.
- Communication in synchronous rounds.
- Each round neighbors exchange $\tilde{O}(1)$-bit msgs.

Model

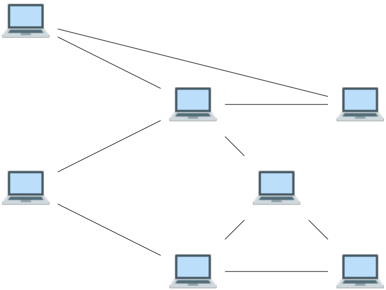


Your favorite large network G.

CONGEST model

- Network topology is an undirected graph.
- Communication in synchronous rounds.
- Each round neighbors exchange $\tilde{O}(1)$-bit msgs.
- Computation is free.

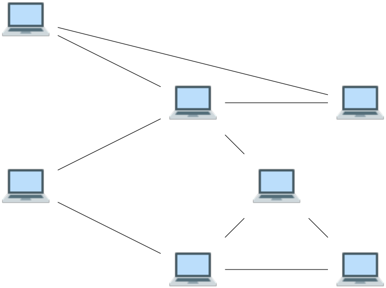
Model



Your favorite large network G.

CONGEST model

- Network topology is an undirected graph.
- Communication in synchronous rounds.
- Each round neighbors exchange $\tilde{O}(1)$-bit msgs.
- Computation is free.
- Initially, nodes know only their immediate neighborhood.



Your favorite large network G.

CONGEST model

- Network topology is an undirected graph.
- Communication in synchronous rounds.
- Each round neighbors exchange $\tilde{O}(1)$-bit msgs.
- Computation is free.
- Initially, nodes know only their immediate neighborhood.
- Objective: minimize # rounds.

Background for distributed MST

	Graph	# rounds

Background for distributed MST

	Graph	# rounds
[GHS 1983]	General graphs	$O(n \log n)$

Background for distributed MST

	Graph	**# rounds**
[GHS 1983]	General graphs	$O(n \log n)$
[Awerbuch 1987]	General graphs	$O(n)$

Background for distributed MST

	Graph	# rounds
[GHS 1983]	General graphs	$O(n \log n)$
[Awerbuch 1987]	General graphs	$O(n)$
Folklore	General graphs	$\Omega(D)$

Background for distributed MST

	Graph	# rounds
[GHS 1983]	General graphs	$O(n \log n)$
[Awerbuch 1987]	General graphs	$O(n)$
Folklore	General graphs	$\Omega(D)$
[GKP 1993]	General graphs	$\tilde{O}(n^{0.613} + D)$

Background for distributed MST

	Graph	# rounds
[GHS 1983]	General graphs	$O(n \log n)$
[Awerbuch 1987]	General graphs	$O(n)$
Folklore	General graphs	$\Omega(D)$
[GKP 1993]	General graphs	$\tilde{O}(n^{0.613} + D)$
[KP 1998]	General graphs	$\tilde{O}(\sqrt{n} + D)$

Background for distributed MST

	Graph	# rounds
[GHS 1983]	General graphs	$O(n \log n)$
[Awerbuch 1987]	General graphs	$O(n)$
Folklore	General graphs	$\Omega(D)$
[GKP 1993]	General graphs	$\tilde{O}(n^{0.613} + D)$
[KP 1998]	General graphs	$\tilde{O}(\sqrt{n} + D)$
[PR 2000]	Worst-case graph	$\tilde{\Omega}(\sqrt{n})$

Background for distributed MST

	Graph	# rounds
[GHS 1983]	General graphs	$O(n \log n)$
[Awerbuch 1987]	General graphs	$O(n)$
Folklore	General graphs	$\Omega(D)$
[GKP 1993]	General graphs	$\tilde{O}(n^{0.613} + D)$
[KP 1998]	General graphs	$\tilde{O}(\sqrt{n} + D)$
[PR 2000]	Worst-case graph	$\tilde{\Omega}(\sqrt{n})$
[GH 2015]	Planar graphs	$\tilde{O}(D)$

Background for distributed MST

	Graph	# rounds
[GHS 1983]	General graphs	$O(n \log n)$
[Awerbuch 1987]	General graphs	$O(n)$
Folklore	General graphs	$\Omega(D)$
[GKP 1993]	General graphs	$\tilde{O}(n^{0.613} + D)$
[KP 1998]	General graphs	$\tilde{O}(\sqrt{n} + D)$
[PR 2000]	Worst-case graph	$\tilde{\Omega}(\sqrt{n})$
[GH 2015]	Planar graphs	$\tilde{O}(D)$
[HI**Z** 2016a/b]	Genus-bounded, treewidth-bounded	$\tilde{O}(D)$

Background for distributed MST

	Graph	# rounds
[GHS 1983]	General graphs	$O(n \log n)$
[Awerbuch 1987]	General graphs	$O(n)$
Folklore	General graphs	$\Omega(D)$
[GKP 1993]	General graphs	$\tilde{O}(n^{0.613} + D)$
[KP 1998]	General graphs	$\tilde{O}(\sqrt{n} + D)$
[PR 2000]	Worst-case graph	$\tilde{\Omega}(\sqrt{n})$
[GH 2015]	Planar graphs	$\tilde{O}(D)$
[HI**Z** 2016a/b]	Genus-bounded, treewidth-bounded	$\tilde{O}(D)$
[HL**Z** 2018]	Minor-free	$\tilde{O}(D^2)$

Background for distributed MST

	Graph	# rounds
[GHS 1983]	General graphs	$O(n \log n)$
[Awerbuch 1987]	General graphs	$O(n)$
Folklore	General graphs	$\Omega(D)$
[GKP 1993]	General graphs	$\tilde{O}(n^{0.613} + D)$
[KP 1998]	General graphs	$\tilde{O}(\sqrt{n} + D)$
[PR 2000]	Worst-case graph	$\tilde{\Omega}(\sqrt{n})$
[GH 2015]	Planar graphs	$\tilde{O}(D)$
[HI**Z** 2016a/b]	Genus-bounded, treewidth-bounded	$\tilde{O}(D)$
[HL**Z** 2018]	Minor-free	$\tilde{O}(D^2)$
[GH 2020]	Minor-free	$\tilde{O}(D)$

Shortcoming of the current state-of-the-art

Matching bounds only for worst-case G and special graph classes.

Shortcoming of the current state-of-the-art

Matching bounds only for worst-case G and special graph classes.

Open problems [Garay, Kutten, Peleg, 1993]

- What graph parameters characterize the complexity of distributed MST (and other problems)?
- Are there universally-optimal algorithms that are as fast as possible on every topology?

Shortcoming of the current state-of-the-art

Matching bounds only for worst-case G and special graph classes.

Open problems [Garay, Kutten, Peleg, 1993]

- What graph parameters characterize the complexity of distributed MST (and other problems)?
- Are there universally-optimal algorithms that are as fast as possible on every topology?

We answer both of these questions.

Our results

For every undirected graph G there is a graph parameter **ShortcutQuality**(G) [Ghaffari, Haeupler, 2015].

For every undirected graph G there is a graph parameter **ShortcutQuality**(G) [Ghaffari, Haeupler, 2015].

- Lower bound (impossibility view):

Theorem

Distributed MST requires at least $\tilde{\Omega}(\textbf{\textit{ShortcutQuality}}(G))$ *time.*

Our results

For every undirected graph G there is a graph parameter **ShortcutQuality**(G) [Ghaffari, Haeupler, 2015].

- Lower bound (impossibility view):

Theorem

Distributed MST requires at least $\tilde{\Omega}(\textbf{ShortcutQuality}(G))$ time.

- Upper bound (algorithmic view):

Theorem

Distributed MST can be solved in $\tilde{O}(\textbf{ShortcutQuality}(G))$ time if the topology is known in advance (but not the input!).

Consequences 1/2

- New result: universal optimality = as fast as possible on every network.
  - Intuition: “perfectly adapts to the network!”
  - We achieve it in the known-topology setting!

Consequences 1/2

- New result: universal optimality = as fast as possible on every network.
  - Intuition: "perfectly adapts to the network!"
  - We achieve it in the known-topology setting!
- Old notion: "existential optimality" = optimal in a class of graphs.
  - Depends on the parameterization ($\tilde{O}(\sqrt{n} + D)$ is optimal only when parameterizing via n and D).
  - Universal optimality is optimal with respect to all parameterizations.

- Same method works for many other problems:
  - (Approx) distributed SSSP.
  - (Approx) distributed mincut.
  - Distributed connectivity verification.
- Moreover, $\tilde{O}(\textbf{ShortcutQuality}(G))$ characterizes all of them.

Consequences 2/2

- Same method works for many other problems:
  - (Approx) distributed SSSP.
  - (Approx) distributed mincut.
  - Distributed connectivity verification.
- Moreover, $\tilde{O}(\textbf{ShortcutQuality}(G))$ characterizes all of them.
- These problems inter-reduce to each other.

Shortcut definition

Definition ([Ghaffari, Haeupler, 2015])

In a graph $G = (V, E)$ we are given connected node-disjoint parts $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$, $P_i \subseteq V$. A shortcut of quality Q for $\mathcal{P}$ is:

Shortcut definition

Definition ([Ghaffari, Haeupler, 2015])

In a graph $G = (V, E)$ we are given connected node-disjoint parts $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$, $P_i \subseteq V$. A shortcut of quality Q for $\mathcal{P}$ is:

1. (Shortcut edges) Each part P_i gets a set of edges $F_i \subseteq E$.

Shortcut definition

Definition ([Ghaffari, Haeupler, 2015])

In a graph $G = (V, E)$ we are given connected node-disjoint parts $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$, $P_i \subseteq V$. A shortcut of quality Q for $\mathcal{P}$ is:

1. (Shortcut edges) Each part P_i gets a set of edges $F_i \subseteq E$.
2. (Dilation) The diameter of $G[P_i] + G[F_i]$ is at most Q.

Shortcut definition

Definition ([Ghaffari, Haeupler, 2015])

In a graph $G = (V, E)$ we are given connected node-disjoint parts $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$, $P_i \subseteq V$. A shortcut of quality Q for $\mathcal{P}$ is:

1. (Shortcut edges) Each part P_i gets a set of edges $F_i \subseteq E$.
2. (Dilation) The diameter of $G[P_i] + G[F_i]$ is at most Q.
3. (Congestion) Each edge $e \in E$ is used by at most Q different F_i's.

Shortcut definition

Definition ([Ghaffari, Haeupler, 2015])

In a graph $G = (V, E)$ we are given connected node-disjoint parts $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$, $P_i \subseteq V$. A shortcut of quality Q for $\mathcal{P}$ is:

1. (Shortcut edges) Each part P_i gets a set of edges $F_i \subseteq E$.
2. (Dilation) The diameter of $G[P_i] + G[F_i]$ is at most Q.
3. (Congestion) Each edge $e \in E$ is used by at most Q different F_i's.

Definition

$$\textbf{ShortcutQuality}(G) = \max_{\mathcal{P}} \min_{\text{shortcut for } \mathcal{P}} \mathrm{quality}(\mathcal{P})$$

Shortcut application: part-wise aggregation

Example (Part-wise aggregation [Ghaffari, Haeupler, 2015])

We are given connected node-disjoint parts $\{P_1, P_2, \ldots, P_k\}$. Each node v has a $O(\log n)$-bit private input x_v. Each part needs to learn the minimum of the inputs in it.

Shortcut application: part-wise aggregation

Example (Part-wise aggregation [Ghaffari, Haeupler, 2015])

We are given connected node-disjoint parts $\{P_1, P_2, \ldots, P_k\}$. Each node v has a $O(\log n)$-bit private input x_v. Each part needs to learn the minimum of the inputs in it.

Lemma ([Ghaffari, Haeupler, 2015])

Given a quality-Q shortcut on $\{P_1, \ldots, P_k\}$, we can solve the part-wise aggregation problem in $\tilde{O}(Q)$ rounds.

Hints on solving part-wise aggregation via quality-Q shortcuts:

- Assume each part P_i has a leader $v_i \in P_i$ (easy exercise).
- All parts concurrently build a BFS tree of $H_i := G[P_i] + G[F_i]$:
  - The leader v_i becomes "active" in a uniformly random time $\{0, \ldots, Q\}$.
  - When a node becomes active, it spreads a message along its neighbors in H_i (only once).
  - A node becomes active the first time it hears a message from part i.
  - Analysis: in every round at most $O(\log n)$ messages are scheduled to pass through an edge, with high probability. We send those messages by subdividing each round into $O(\log n)$ subrounds. Since the BFS tree has depth Q, the process completes in $O(Q \log n)$ subdivided rounds.
- Spread the maximum using the BFS tree using the same idea (randomly delay each part by $\{0, 1, \ldots, Q\}$ and flood-fill the tree).

Shortcut application: MST

Definition (Construction oracle)

Suppose that for each set of connected and node-disjoint parts $\{P_1, \ldots, P_k\}$ we can construct a shortcut of quality Q.

Shortcut application: MST

Definition (Construction oracle)

Suppose that for each set of connected and node-disjoint parts $\{P_1, \ldots, P_k\}$ we can construct a shortcut of quality Q.

Example ([Ghaffari, Haeupler, 2015])

Given a construction oracle of quality Q, we can solve MST in $\tilde{O}(Q)$ rounds.

Proof. Run Boruvka's algorithm.

Shortcut application: MST

Definition (Construction oracle)

Suppose that for each set of connected and node-disjoint parts $\{P_1, \ldots, P_k\}$ we can construct a shortcut of quality Q.

Example ([Ghaffari, Haeupler, 2015])

Given a construction oracle of quality Q, we can solve MST in $\tilde{O}(Q)$ rounds.

Proof. Run Boruvka's algorithm.

- Each node v finds the minimum outgoing edge.

Shortcut application: MST

Definition (Construction oracle)

Suppose that for each set of connected and node-disjoint parts $\{P_1, \ldots, P_k\}$ we can construct a shortcut of quality Q.

Example ([Ghaffari, Haeupler, 2015])

Given a construction oracle of quality Q, we can solve MST in $\tilde{O}(Q)$ rounds.

Proof. Run Boruvka's algorithm.

- Each node v finds the minimum outgoing edge.
- Add that edge to the MST.

Shortcut application: MST

Definition (Construction oracle)

Suppose that for each set of connected and node-disjoint parts $\{P_1, \ldots, P_k\}$ we can construct a shortcut of quality Q.

Example ([Ghaffari, Haeupler, 2015])

Given a construction oracle of quality Q, we can solve MST in $\tilde{O}(Q)$ rounds.

Proof. Run Boruvka's algorithm.

- Each node v finds the minimum outgoing edge.
- Add that edge to the MST.
- Contract that edge.

Shortcut application: MST

Definition (Construction oracle)

Suppose that for each set of connected and node-disjoint parts $\{P_1, \ldots, P_k\}$ we can construct a shortcut of quality Q.

Example ([Ghaffari, Haeupler, 2015])

Given a construction oracle of quality Q, we can solve MST in $\tilde{O}(Q)$ rounds.

Proof. Run Boruvka's algorithm.

- Each node v finds the minimum outgoing edge.
- Add that edge to the MST.
- Contract that edge.
- Repeat $O(\log n)$ times.

1. Introduction

2. Preliminary: Shortcuts

3. High-level technical overview
   - Upper bound: shortcut construction
   - Lower bound: distributed disjointness task
   - Lower bound: ingredients

4. Lower bound: more details

5. Conclusion and Open Questions

Upper bound: shortcut construction

Question: Can we efficiently construct shortcuts?

Answer: Yes! But currently only in the known topology setting.

Upper bound: shortcut construction

Question: Can we efficiently construct shortcuts?

Answer: Yes! But currently only in the known topology setting.

- Challenge 1: parts need to construct shortcuts without learning (much) about other parts.

Upper bound: shortcut construction

Question: Can we efficiently construct shortcuts?

Answer: Yes! But currently only in the known topology setting.

- Challenge 1: parts need to construct shortcuts without learning (much) about other parts.
  - Challenge even in the known topology setting.

Question: Can we efficiently construct shortcuts?

Answer: Yes! But currently only in the known topology setting.

- Challenge 1: parts need to construct shortcuts without learning (much) about other parts.
  - Challenge even in the known topology setting.
  - Solution: Oblivious routing!

Upper bound: shortcut construction

Question: Can we efficiently construct shortcuts?

Answer: Yes! But currently only in the known topology setting.

- Challenge 1: parts need to construct shortcuts without learning (much) about other parts.
  - Challenge even in the known topology setting.
  - Solution: Oblivious routing!
- Challenge 2: we need to balance between both diameter and congestion.

Upper bound: shortcut construction

Question: Can we efficiently construct shortcuts?

Answer: Yes! But currently only in the known topology setting.

- Challenge 1: parts need to construct shortcuts without learning (much) about other parts.
  - Challenge even in the known topology setting.
  - Solution: Oblivious routing!
- Challenge 2: we need to balance between both diameter and congestion.
  - Standard solutions fail.

Upper bound: shortcut construction

Question: Can we efficiently construct shortcuts?

Answer: Yes! But currently only in the known topology setting.

- Challenge 1: parts need to construct shortcuts without learning (much) about other parts.
  - Challenge even in the known topology setting.
  - Solution: Oblivious routing!
- Challenge 2: we need to balance between both diameter and congestion.
  - Standard solutions fail.
  - Solution: see the talk "Hop-Constrained Oblivious Routing" [GH**Z** STOC'21] on Youtube.

Upper bound: shortcut construction

Question: Can we efficiently construct shortcuts?

Answer: Yes! But currently only in the known topology setting.

- Challenge 1: parts need to construct shortcuts without learning (much) about other parts.
  - Challenge even in the known topology setting.
  - Solution: Oblivious routing!
- Challenge 2: we need to balance between both diameter and congestion.
  - Standard solutions fail.
  - Solution: see the talk "Hop-Constrained Oblivious Routing" [GH**Z** STOC'21] on Youtube.

Theorem (Upper bound)

Suppose all nodes know the topology G upfront. We can construct shortcuts of near-optimal quality Q in $\tilde{O}(Q)$ rounds.

Lower bound: distributed disjointness task

Distributed disjointness task

- Alice and Bob have k-bit inputs x and y, resp.

Lower bound: distributed disjointness task

Distributed disjointness task

- Alice and Bob have k-bit inputs x and y, resp.
- We are given k node-disjoint paths $P_1, \ldots, P_k$.

Lower bound: distributed disjointness task

Distributed disjointness task

- Alice and Bob have k-bit inputs x and y, resp.
- We are given k node-disjoint paths $P_1, \ldots, P_k$.
- Alice controls the heads of the paths $S = \{s_1, \ldots, s_k\}$; Bob controls the tails $T = \{t_1, \ldots, t_k\}$.

Lower bound: distributed disjointness task

Distributed disjointness task

- Alice and Bob have k-bit inputs x and y, resp.
- We are given k node-disjoint paths $P_1, \ldots, P_k$.
- Alice controls the heads of the paths $S = \{s_1, \ldots, s_k\}$; Bob controls the tails $T = \{t_1, \ldots, t_k\}$.
- What is the minimum amount of rounds until Alice/Bob decide whether $\exists i \in \{1, \ldots, k\}$ such that $x_i = 1$ and $y_i = 1$.

Lower bound: distributed disjointness task

Distributed disjointness task

- Alice and Bob have k-bit inputs x and y, resp.
- We are given k node-disjoint paths $P_1, \ldots, P_k$.
- Alice controls the heads of the paths $S = \{s_1, \ldots, s_k\}$; Bob controls the tails $T = \{t_1, \ldots, t_k\}$.
- What is the minimum amount of rounds until Alice/Bob decide whether $\exists i \in \{1, \ldots, k\}$ such that $x_i = 1$ and $y_i = 1$.

Theorem

The distributed disjointness task on each subset of paths $\{P_1, \ldots, P_k\}$ can be solved in Q rounds.

if and only if

There exists shortcut of quality $\tilde{O}(Q)$ for $P_1, \ldots, P_k$.

- See "Network Coding Gaps for Completion Times of Multiple Unicasts" [HWZ FOCS'20] on Youtube.

Lower bound: ingredients

We want to prove:

Theorem

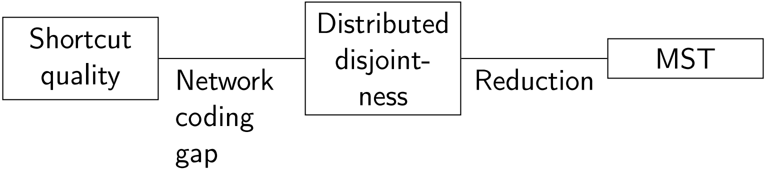
$$T_{MST}(G) \geq \tilde{\Omega}(1) \cdot \mathrm{ShortcutQuality}(G).$$

Lower bound: ingredients

We want to prove:

Theorem

$$T_{MST}(G) \geq \tilde{\Omega}(1) \cdot \mathrm{ShortcutQuality}(G).$$



Lower bound: statement

We want to prove:

Theorem

$$T_{MST}(G) \geq \tilde{\Omega}(1) \cdot \mathrm{ShortcutQuality}(G).$$

We want to prove:

Theorem

$$T_{MST}(G) \geq \tilde{\Omega}(1) \cdot \mathrm{ShortcutQuality}(G).$$

Equivalent:

Lemma

Given any set of connected and node-disjoint parts $P_1, \ldots, P_k$ we can construct shortcuts of quality $\tilde{O}(T_{MST})$ on them.

We want to prove:

Theorem

$$T_{MST}(G) \geq \tilde{\Omega}(1) \cdot \mathrm{ShortcutQuality}(G).$$

Equivalent:

Lemma

Given any set of connected and node-disjoint parts $P_1, \ldots, P_k$ we can construct shortcuts of quality $\tilde{O}(T_{MST})$ on them.

Equivalent:

Lemma

Given any set of node-disjoint paths $P_1, \ldots, P_k$ we can construct shortcuts of quality $\tilde{O}(T_{MST})$ on them.

Hints on why is it sufficient to consider only node-disjoint paths P_i (instead of arbitrary connected and node-disjoint subsets):

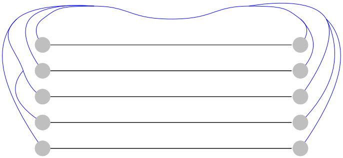
- Let $T_1, \ldots, T_k$ be some spanning trees of $P_1, \ldots, P_k$ (note: P_i is connected).
- Root each T_i and consider the heavy-light decomposition of T_i, which decomposes any tree into a number of paths such that for any path p there are at most HL-depth(p) $\leq O(\log n)$ paths on the root-to-p path in T_i.
- For $i = O(\log n)$ down to 1 do:
  - Consider all paths of HL-depth(p) $= i$.
  - By assumption, we can construct shortcuts of quality $\tilde{O}(Q)$ on them (each path is its own part). Construct it.
- The shortcut of P_i is the union of the shortcuts associated paths of the heavy-light decomposition of T_i.
- Since the shortcuts of P_i's were constructed by $O(\log n)$ calls to the path-wise shortcut oracle, their quality increases by a negligible $O(\log n)$ compared to the shortcuts of the paths.

Disjointness gadget: definition

Definition

A disjointness gadget of a set of node-disjoint paths $P_1, \ldots, P_k$ is a connected subset of edges F that touches the heads/tails of each path, but does not otherwise intersect the interior.[1]

[1] We also allow $O(1)$ "exception intervals" of length $O(D)$ on each P_i where F can intersect.

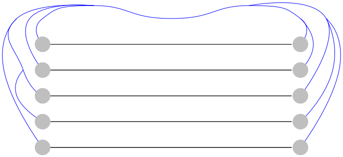


A disjointness gadget

Observation

Let F be a disjointness gadget of node-disjoint paths $\mathcal{P}$. Using a single call to the MST oracle, we can solve the distributed disjointness task on $\mathcal{P}$.

Idea: Given Alice/Bob inputs x, y we assign MST costs such that MST has cost 0 if and only if x and y are disjoint.



Theorem (Main technical contribution of the paper)

Given any set of node-disjoint paths P, there exists a disjointness gadget on a subset $P' \subseteq P$ of size $|P'| \geq \frac{1}{\text{poly}(\log n)}|P|$.

Completing the proof:

- It is sufficient to construct of quality T_{MST} on arbitrary node-disjoint paths $\mathcal{P}$.

Theorem (Main technical contribution of the paper)

Given any set of node-disjoint paths P, there exists a disjointness gadget on a subset $P' \subseteq P$ of size $|P'| \geq \frac{1}{\text{poly}(\log n)}|P|$.

Completing the proof:

- It is sufficient to construct of quality T_{MST} on arbitrary node-disjoint paths $\mathcal{P}$.
- Find a disjointness gadget on a large subset $\mathcal{P}' \subseteq \mathcal{P}$.

Theorem (Main technical contribution of the paper)

Given any set of node-disjoint paths P, there exists a disjointness gadget on a subset $P' \subseteq P$ of size $|P'| \geq \frac{1}{\text{poly}(\log n)}|P|$.

Completing the proof:

- It is sufficient to construct of quality T_{MST} on arbitrary node-disjoint paths $\mathcal{P}$.
- Find a disjointness gadget on a large subset $\mathcal{P}' \subseteq \mathcal{P}$.
- We can solve the distributed disjointness task on $\mathcal{P}'$ in $\tilde{O}(T_{MST})$ time.

Theorem (Main technical contribution of the paper)

Given any set of node-disjoint paths P, there exists a disjointness gadget on a subset $P' \subseteq P$ of size $|P'| \geq \frac{1}{\text{poly}(\log n)}|P|$.

Completing the proof:

- It is sufficient to construct of quality T_{MST} on arbitrary node-disjoint paths $\mathcal{P}$.
- Find a disjointness gadget on a large subset $\mathcal{P}' \subseteq \mathcal{P}$.
- We can solve the distributed disjointness task on $\mathcal{P}'$ in $\tilde{O}(T_{MST})$ time.
- Via network coding gap, there exists shortcut on $\mathcal{P}'$ of quality $\tilde{O}(T_{MST})$.

Theorem (Main technical contribution of the paper)

Given any set of node-disjoint paths P, there exists a disjointness gadget on a subset $P' \subseteq P$ of size $|P'| \geq \frac{1}{\text{poly}(\log n)}|P|$.

Completing the proof:

- It is sufficient to construct of quality T_{MST} on arbitrary node-disjoint paths $\mathcal{P}$.
- Find a disjointness gadget on a large subset $\mathcal{P}' \subseteq \mathcal{P}$.
- We can solve the distributed disjointness task on $\mathcal{P}'$ in $\tilde{O}(T_{MST})$ time.
- Via network coding gap, there exists shortcut on $\mathcal{P}'$ of quality $\tilde{O}(T_{MST})$.
- Remove $\mathcal{P}'$ from $\mathcal{P}$ and repeat $\tilde{O}(1)$ times. The final shortcut is still of quality $\tilde{O}(T_{MST})$.

Disjointness gadget: construction

Theorem (Main technical contribution of the paper)

Given any set of node-disjoint paths P, there exists a disjointness gadget on a subset $P' \subseteq P$ of size $|P'| \geq \frac{1}{\text{poly}(\log n)}|P|$.

Theorem (Simplified construction)

Given any set of node-disjoint paths P, there exists a disjointness gadget on a subset $P' \subseteq P$ of size $|P'| \geq \frac{1}{O(D)}|P|$.

Disjointness gadget: construction

Theorem (Simplified construction)

Given any set of node-disjoint paths P, there exists a disjointness gadget on a subset $P' \subseteq P$ of size $|P'| \geq \frac{1}{O(D)}|P|$.

- Choose an arbitrary "root" r.

Disjointness gadget: construction

Theorem (Simplified construction)

Given any set of node-disjoint paths P, there exists a disjointness gadget on a subset $P' \subseteq P$ of size $|P'| \geq \frac{1}{O(D)}|P|$.

- Choose an arbitrary "root" r.
- Consider adding p_i to P'.

Disjointness gadget: construction

Theorem (Simplified construction)

Given any set of node-disjoint paths P, there exists a disjointness gadget on a subset $P' \subseteq P$ of size $|P'| \geq \frac{1}{O(D)}|P|$.

- Choose an arbitrary "root" r.
- Consider adding p_i to P'.
  - Walk from head/tail to the root.

Disjointness gadget: construction

Theorem (Simplified construction)

Given any set of node-disjoint paths P, there exists a disjointness gadget on a subset $P' \subseteq P$ of size $|P'| \geq \frac{1}{O(D)}|P|$.

- Choose an arbitrary “root” r.
- Consider adding p_i to P'.
  - Walk from head/tail to the root.
  - Add this walk to F.

Disjointness gadget: construction

Theorem (Simplified construction)

Given any set of node-disjoint paths P, there exists a disjointness gadget on a subset $P' \subseteq P$ of size $|P'| \geq \frac{1}{O(D)}|P|$.

- Choose an arbitrary "root" r.
- Consider adding p_i to P'.
  - Walk from head/tail to the root.
  - Add this walk to F.
  - p_i "deletes" all paths it encounters.

Disjointness gadget: construction

Theorem (Simplified construction)

Given any set of node-disjoint paths P, there exists a disjointness gadget on a subset $P' \subseteq P$ of size $|P'| \geq \frac{1}{O(D)}|P|$.

- Choose an arbitrary "root" r.
- Consider adding p_i to P'.
  - Walk from head/tail to the root.
  - Add this walk to F.
  - p_i "deletes" all paths it encounters.
  - Self-intersecting parts are exceptional intervals.

Disjointness gadget: construction

Theorem (Simplified construction)

Given any set of node-disjoint paths P, there exists a disjointness gadget on a subset $P' \subseteq P$ of size $|P'| \geq \frac{1}{O(D)}|P|$.

- Choose an arbitrary "root" r.
- Consider adding p_i to P'.
  - Walk from head/tail to the root.
  - Add this walk to F.
  - p_i "deletes" all paths it encounters.
  - Self-intersecting parts are exceptional intervals.
- Each p_i deletes $O(D)$ other paths.

Disjointness gadget: construction

Theorem (Simplified construction)

Given any set of node-disjoint paths P, there exists a disjointness gadget on a subset $P' \subseteq P$ of size $|P'| \geq \frac{1}{O(D)}|P|$.

- Choose an arbitrary "root" r.
- Consider adding p_i to P'.
  - Walk from head/tail to the root.
  - Add this walk to F.
  - p_i "deletes" all paths it encounters.
  - Self-intersecting parts are exceptional intervals.
- Each p_i deletes $O(D)$ other paths.
- So, there must exist an independent subset $|P'| \geq \frac{1}{O(D)}|P|$.

Conclusion and Open Questions

- First universal lower bound for problems like distributed MST.

Conclusion and Open Questions

- First universal lower bound for problems like distributed MST.
- First universally-optimal algorithms (when the topology is known).

Conclusion and Open Questions

- First universal lower bound for problems like distributed MST.
- First universally-optimal algorithms (when the topology is known).
- Conjecture: shortcuts can be constructed efficiently $\implies$ characterization in unknown topology.

Conclusion and Open Questions

- First universal lower bound for problems like distributed MST.
- First universally-optimal algorithms (when the topology is known).
- Conjecture: shortcuts can be constructed efficiently $\implies$ characterization in unknown topology.
- Connections to many other fields of TCS.

- First universal lower bound for problems like distributed MST.
- First universally-optimal algorithms (when the topology is known).
- Conjecture: shortcuts can be constructed efficiently $\implies$ characterization in unknown topology.
- Connections to many other fields of TCS.
  - New network coding gaps.
  - New types of oblivious routings.
  - New connections between distributed computing and communication complexity.

Open questions:

Conclusion and Open Questions

- First universal lower bound for problems like distributed MST.
- First universally-optimal algorithms (when the topology is known).
- Conjecture: shortcuts can be constructed efficiently $\implies$ characterization in unknown topology.
- Connections to many other fields of TCS.
  - New network coding gaps.
  - New types of oblivious routings.
  - New connections between distributed computing and communication complexity.

Open questions:

- Universal optimality in other models?
- Universal optimality for other problems?

Conclusion and Open Questions

- First universal lower bound for problems like distributed MST.
- First universally-optimal algorithms (when the topology is known).
- Conjecture: shortcuts can be constructed efficiently $\implies$ characterization in unknown topology.
- Connections to many other fields of TCS.
  - New network coding gaps.
  - New types of oblivious routings.
  - New connections between distributed computing and communication complexity.

Open questions:

- Universal optimality in other models?
- Universal optimality for other problems?

Thank you!