Low-Congestion Shortcuts

Routing for Distributed Optimization Algorithms

Bernhard Haeupler

TUM → MIT → CMU

Based on tons of prior work and joint work with

Mohsen Ghaffari, MIT → ETH

Goran Zuzic & Jason Li, CMU

Taisuke Izumi, Nagoya IT

supported through NSF award “Distributed Algorithms for Near Planar Networks”

Theme of this Talk

The bottleneck in most distributed computations is communication.

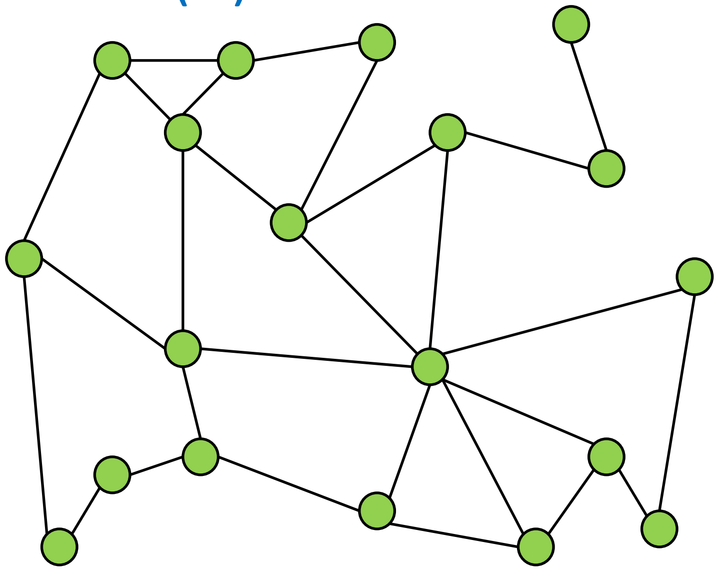
Any distributed optimization algorithm in a bandwidth limited network needs to prioritize the processing and temporal routing of information through the network.

How does this look like and how does one do this efficiently?

Distributed Optimization

Message Passing Model (CONGEST):

- (Weighted) network $G = (V, E)$ with $n = |V|$ machines and D=diam(G).
- Initially each node only knows its incident edges.
- Per round, O(log n) bits can be sent over each edge.
- Local computation is free.



Goal:
Compute an MST, Shortest Path Tree, Min-Cut, etc. while minimizing the number of rounds.

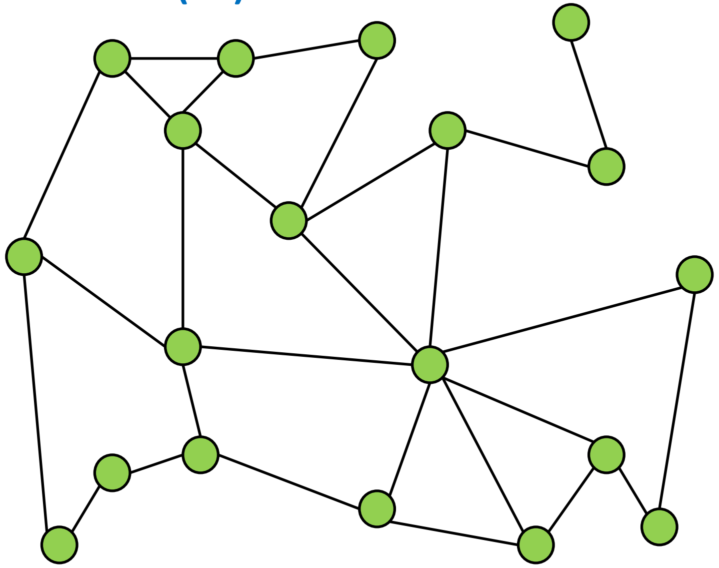
Trivial Round Complexities:

- All non-local problems require $\Omega(D)$ rounds.
- Any problem can be solved in $O(m)$ rounds.
- This is generally a big gap! Think of $D = n^{o(1)}$ or $D = \log^{O(1)} n$.

Distributed Optimization

Message Passing Model (CONGEST):

- (Weighted) network $G = (V, E)$ with $n = |V|$ machines and D=diam(G).
- Initially each node only knows its incident edges.
- Per round, O(log n) bits can be sent over each edge.
- Local computation is free.



Dream Goal:
Compute an MST, Shortest Path Tree, Min-Cut, etc.. in $\tilde{O}(D)$ rounds.

Trivial Round Complexities:

- All non-local problems require $\Omega(D)$ rounds.
- Any problem can be solved in $O(m)$ rounds.
- This is generally a big gap! Think of $D = n^{o(1)}$ or $D = \log^{O(1)} n$.

Minimum Spanning Tree (MST)

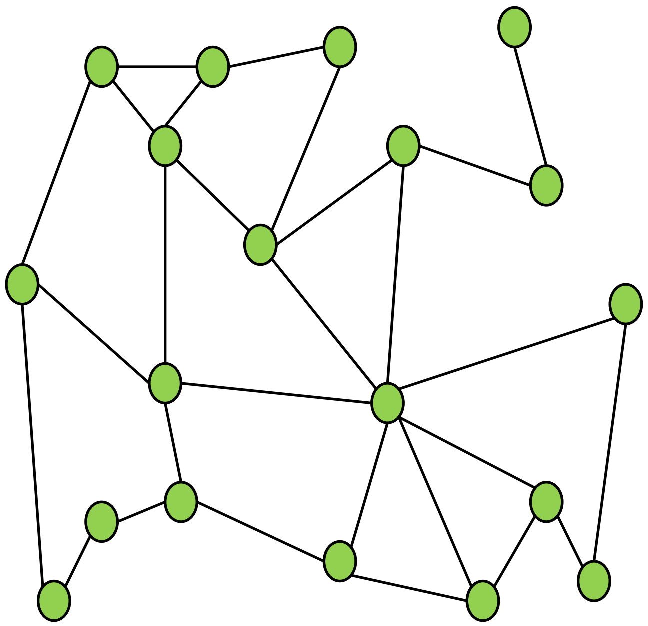
MST Problem: Identify the cheapest set of edges T that form a spanning tree.

Boruvka's MST Algorithm [1926]

Start with T=∅, i.e., with each machine in its own connected component

Repeat until done

Each connected component in $G[T]$ adds its cheapest outgoing edge to T



Minimum Spanning Tree (MST)

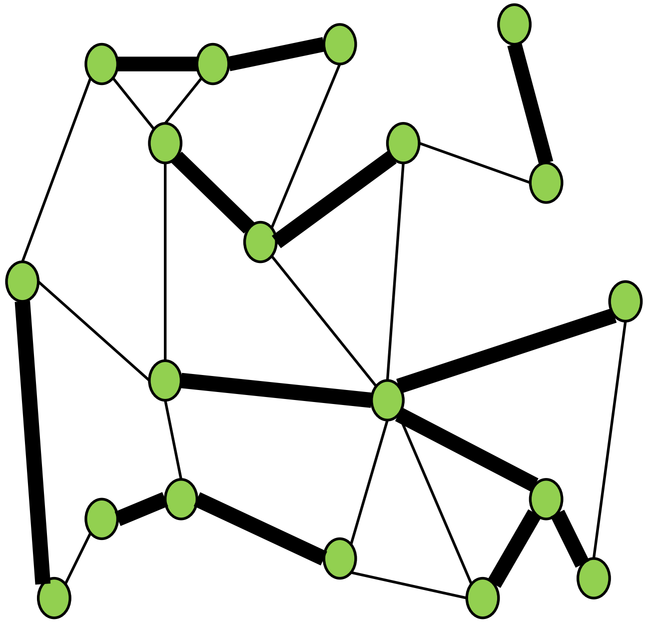
MST Problem: Identify the cheapest set of edges T that form a spanning tree.

Boruvka's MST Algorithm [1926]

Start with T=Ø, i.e., with each machine in its own connected component

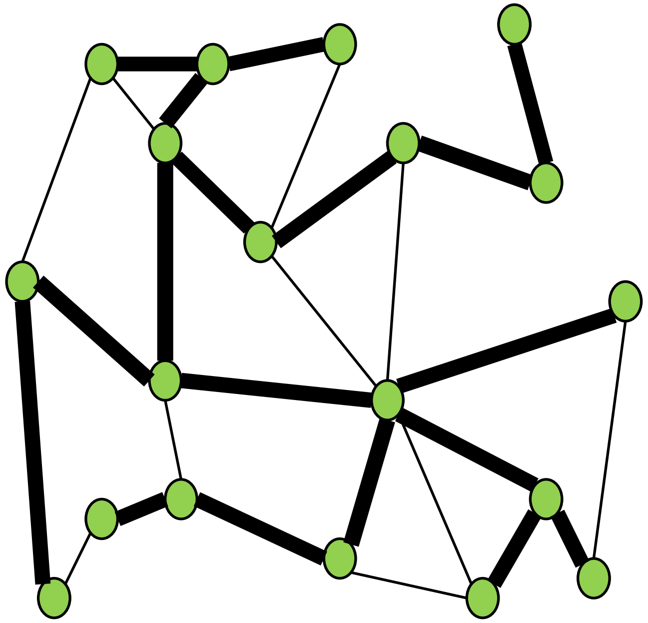
Repeat until done

  Each connected component in $G[T]$ adds its cheapest outgoing edge to T



Minimum Spanning Tree (MST)

MST Problem: Identify the cheapest set of edges T that form a spanning tree.



Boruvka's MST Algorithm [1926]

Start with T=Ø, i.e., with each machine in its own connected component

Repeat until done

  Each connected component in $G[T]$ adds its cheapest outgoing edge to T

$$Round\ complexity\ =\ \#iterations \cdot Time\ per\ iteration$$

Minimum Spanning Tree (MST)

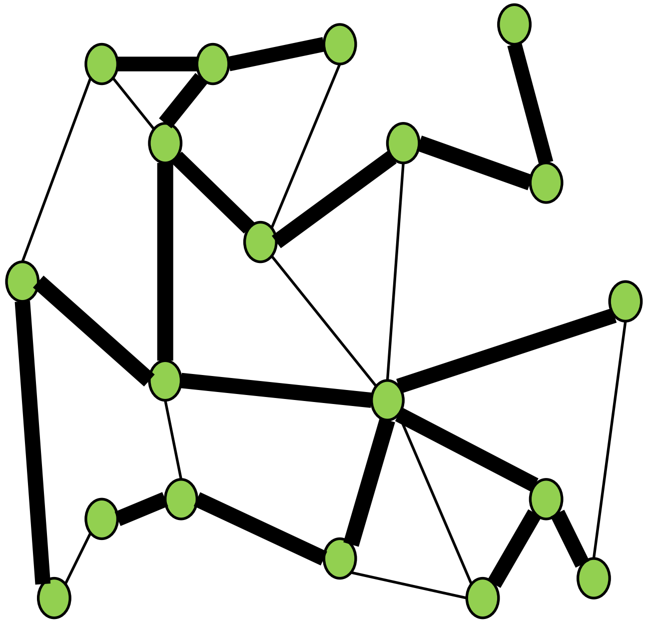
MST Problem: Identify the cheapest set of edges T that form a spanning tree.

Boruvka's MST Algorithm [1926]

Start with T=Ø, i.e., with each machine in its own connected component

Repeat log n times:

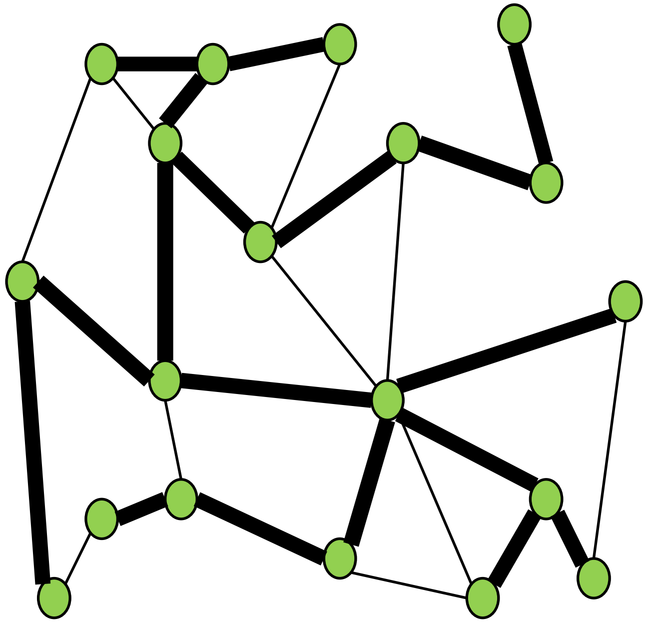
Each connected component in $G[T]$ adds its cheapest outgoing edge to T



$$Round\ complexity\ = \log n \cdot Time\ per\ iteration$$

Minimum Spanning Tree (MST)

MST Problem: Identify the cheapest set of edges T that form a spanning tree.



Boruvka's MST Algorithm [1926]

Start with T=Ø, i.e., with each machine in its own connected component

Repeat log n times:

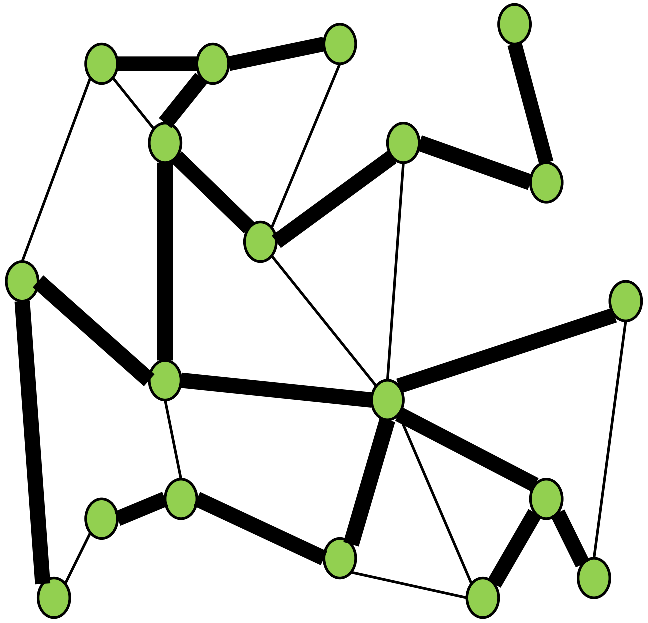
Each connected component in $G[T]$ adds its cheapest outgoing edge to T

$$Round\ complexity\ = \log n \cdot Time\ per\ iteration$$

Identify the cheapest outgoing edge by flooding the minimum weight in each component. This takes at most as many rounds as the diameter.

Minimum Spanning Tree (MST)

MST Problem: Identify the cheapest set of edges T that form a spanning tree.



Boruvka's MST Algorithm [1926]

Start with T=Ø, i.e., with each machine in its own connected component

Repeat log n times:

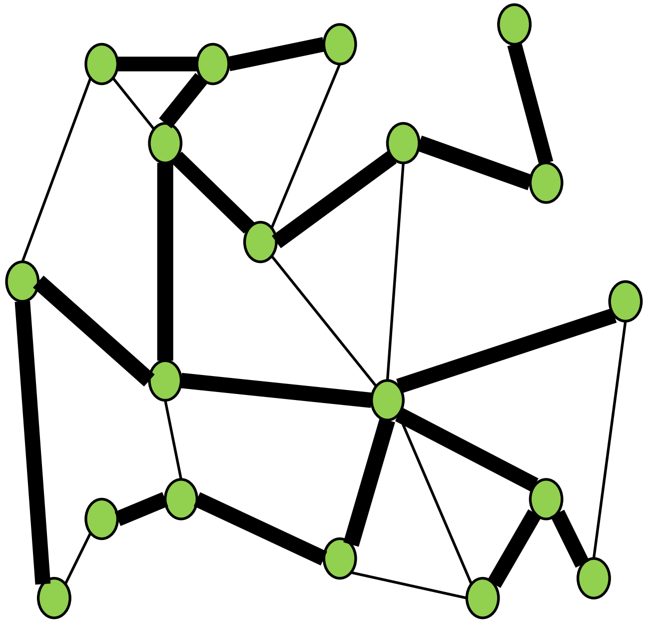
Each connected component in $G[T]$ adds its cheapest outgoing edge to T

$$Round\ complexity\ = \log n \cdot O(D)$$

Identify the cheapest outgoing edge by flooding the minimum weight in each component. This takes at most as many rounds as the diameter.

Minimum Spanning Tree (MST)

MST Problem: Identify the cheapest set of edges T that form a spanning tree.



Boruvka's MST Algorithm [1926]

Start with T=Ø, i.e., with each machine in its own connected component

Repeat log n times:

Each connected component in $G[T]$ adds its cheapest outgoing edge to T

$$Round\ complexity\ = \log n \cdot O(D)$$

Identify the cheapest outgoing edge by flooding the minimum weight in each component. This takes at most as many rounds as the diameter.

Minimum Spanning Tree (MST)

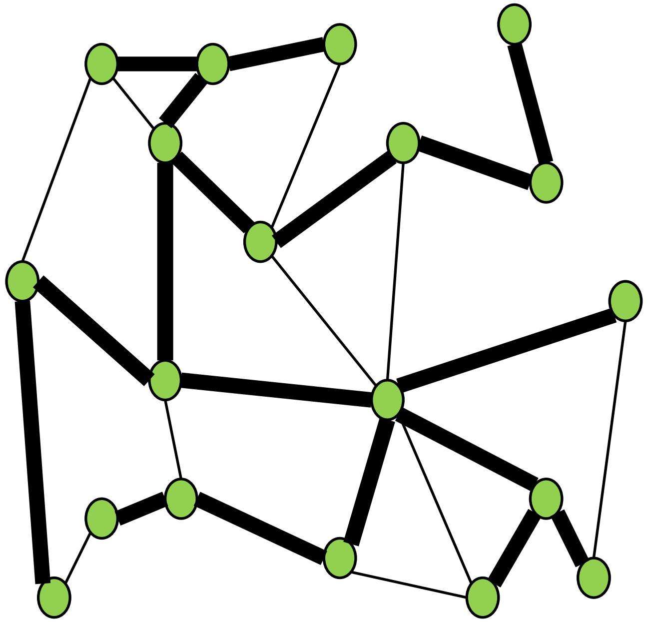
MST Problem: Identify the cheapest set of edges T that form a spanning tree.

Boruvka's MST Algorithm [1926]

Start with T=∅, i.e., with each machine in its own connected component

Repeat log n times:

Each connected component in $G[T]$ adds its cheapest outgoing edge to T



$$Round\ complexity\ = \log n \cdot O(D)$$

Identify the cheapest outgoing edge by flooding the minimum weight in each component. This takes at most as many rounds as the component diameter.

Problem:
Diameter of induced subgraphs can be much larger than the network diameter D!

Minimum Spanning Tree (MST)

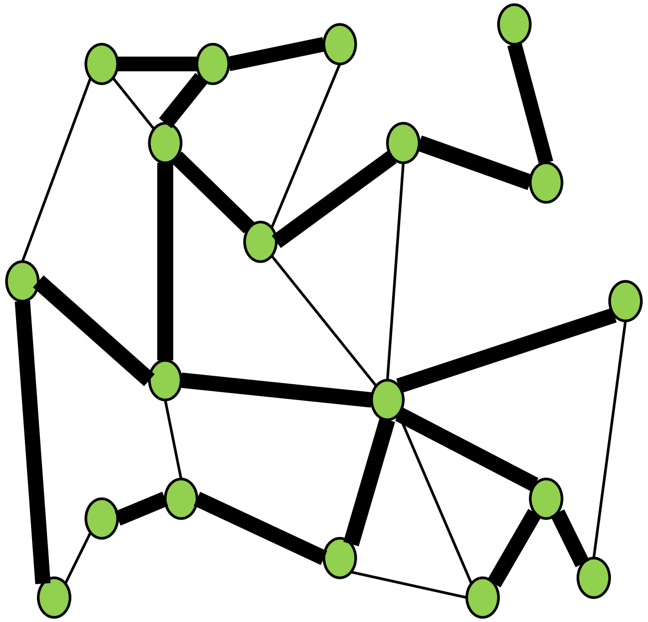
MST Problem: Identify the cheapest set of edges T that form a spanning tree.

Boruvka's MST Algorithm [1926]

Start with T=Ø, i.e., with each machine in its own connected component

Repeat log n times:

Each connected component in $G[T]$ adds its cheapest outgoing edge to T



$$Round\ complexity\ = \log n \cdot O(n)$$

Identify the cheapest outgoing edge by flooding the minimum weight in each component. This takes at most as many rounds as the **component** diameter.

Problem:
Diameter of induced subgraphs can be much larger than the network diameter D!

Distributed Minimum Spanning Tree (prior work)

Celebrated but complicated $\tilde{O}(\sqrt{n} + D)$ algorithm. [KP’95]

Strong $\widetilde{\Omega}(\sqrt{n})$ Lower Bound [RP'99,E'04,DHK+'11]

- holds for almost any non-local network problem
- even for any non-trivial approximation
- despite tiny diameter, e.g., D = log n
- unconditional, based on communication complexity of disjointness

→ We have an “optimal” algorithm (in terms of n and D).

BUT: The lower bound network seems pathological and highly unnatural and the KP algorithm is always $\Omega(\sqrt{n})$ slow including on much nicer networks of interest.

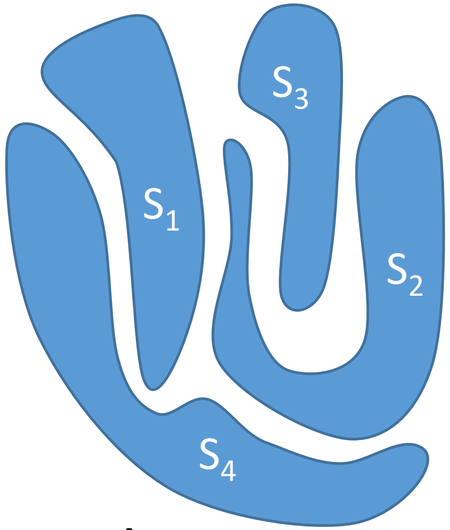

THIS IS WHERE
OUR STORY
BEGINS

Key Problem: Partwise Aggregation

Network G is partitioned into disjoint individually-connected parts S_1, S_2, …, S_N.

Want: Compute a simple (aggregate) function in each part, e.g., min-value.

Challenge: The diameter of parts might be large!



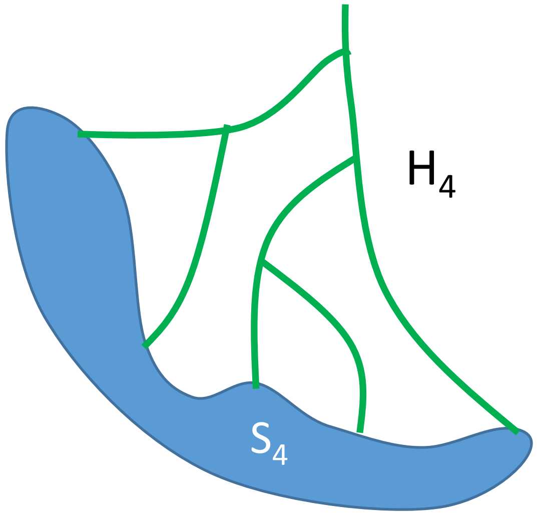
This problem arises naturally in many divide-and-conquer style algorithms.

Informal Claim:

This problem **completely captures** the crux of most distributed optimization problems. I.e., how fast this partwise information aggregation can be solved in a given topology determines approx. how efficient **any** optimization algorithm can be.

Low-Congestion Shortcuts to the Rescue

Idea: Instead of only communication within each part, allow each part to use some shortcut edges & nodes for its communication.



Key definition:

A Shortcut with congestion γ and dilation δ for parts $S_1, S_2, \ldots, S_N$ is

a set of subgraphs $H_1, H_2, \ldots, H_N \subseteq G$, one for each part, such that:

1. $\forall$ part S_i, diameter$(G[S_i]+H_i) \leq \delta$
2. $\forall$ edge e, the number of subgraphs $G[S_i]+H_i$ containing edge e is $\leq \gamma$

Routing & Scheduling in Low-Congestion Shortcuts

Given a shortcut with congestion γ and dilation δ.

How quickly can we solve the partwise aggregation problem?

Routing: Compute a BFS-tree in each $G[S_i]+H_i$ and broadcast / aggregate along the tree.

Remaining Scheduling Problem:
Given many rooted trees of depth $\leq \delta$, s.th., any edge is in at most γ trees. Send a message from each root to its leaves using each edge only once per round. Minimize the makespan.

Routing & Scheduling in Low-Congestion Shortcuts

Scheduling Problem:
Given many rooted trees of depth $\leq \delta$, s.th., any edge is in at most γ trees. Send a message from each root to its leaves using each edge only once per round. Minimize the makespan.

Trivial: $\boldsymbol{O}(\boldsymbol{\delta} \cdot \boldsymbol{\gamma})$ rounds; blow time up by factor of γ and send all messages in parallel.

[LMR'94]: Picking a random delay in $[0, \gamma]$ for each transmission gives:

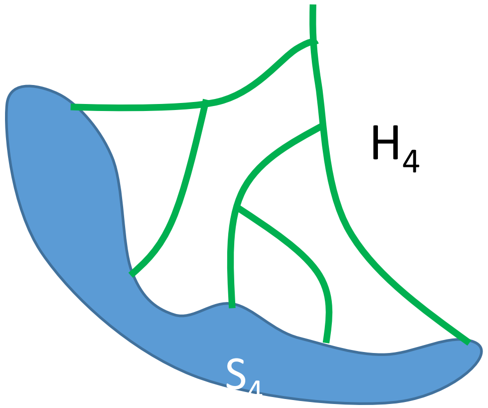
- O(1) expected congestion in each round and $\leq O(\log\ n)$ congestion whp.
- $\delta' \leq \delta + \gamma$

→ There is a simple distributed $\boldsymbol{O}((\boldsymbol{\delta} + \boldsymbol{\gamma}) \cdot\ \boldsymbol{\log n})$ round schedule.

Remark: For simple paths schedules with $O(\delta + \gamma)$ exist. For trees $O(\delta + \gamma + \log^2 n)$ is possible. For DAGs $\Omega((\delta + \gamma) \cdot \frac{\log n}{\log \log n})$ rounds are necessary.

Low-Congestion Shortcuts

Idea: Instead of only communication within each part, allow each part to use some shortcut edges & nodes for its communication.



A Shortcut with congestion $\boldsymbol{\gamma}$ and dilation $\boldsymbol{\delta}$ for parts $S_1, S_2, \ldots, S_N$ is

a set of subgraphs $H_1, H_2, \ldots, H_N \subseteq G$, one for each part, such that:

1. $\forall$ part S_i, diameter$(G[S_i]+H_i) \leq \boldsymbol{\delta}$
2. $\forall$ edge e, the number of subgraphs $G[S_i]+H_i$ containing edge e is $\leq \boldsymbol{\gamma}$

Shortcuts allows to solve the partwise aggregation problem in $\tilde{O}(\gamma + \delta)$ rounds.

→ We mostly care about $\gamma + \delta$, which we call the ***quality*** of a shortcut.

Trivial Low-Congestion Shortcuts

A Shortcut with congestion $\boldsymbol{\gamma}$ and dilation $\boldsymbol{\delta}$ for parts $S_1, S_2, \ldots, S_N$ is

a set of subgraphs $H_1, H_2, \ldots, H_N \subseteq G$, one for each part, such that:

1. $\forall$ part S_i, diameter($G[S_i]+H_i$) $\leq \boldsymbol{\delta}$
2. $\forall$ edge e, the number of subgraphs $G[S_i]+H_i$ containing edge e is $\leq \boldsymbol{\gamma}$

Shortcuts allows to solve the partwise aggregation problem in $\tilde{O}(\gamma + \delta)$ rounds.

Trivial Bounds on γ, δ and quality $Q = \gamma + \delta$:

- Any graph partitioning has a shortcut with $\gamma = 1$, $\delta = n$, and $Q = n$.
- Any graph partitioning has a shortcut with $\gamma \leq n$, $\delta = D$, and $Q = n$.
- Any graph partitioning has a shortcut with $\gamma \leq \sqrt{n}$, $\delta = \sqrt{n} + D$, and $Q = O(\sqrt{n} + D)$.
  (give parts with more than $\sqrt{n}$ nodes all of G)

A Simple $\tilde{\boldsymbol{O}}(\sqrt{\boldsymbol{n}} + \boldsymbol{D})$ MST Algorithm

Boruvka's MST Algorithm with Shortcuts

Start with T=∅

Repeat log n times:

Compute a low congestion shortcut for the connected components

Using the random delay routing, compute the cheapest outgoing edge for each component

Add these edges to T

Running time: $O(Q \log n \cdot \log n) = O((\sqrt{n} + D) \log^2 n)$

Generally no better shortcuts or algorithms are possible.

BUT: This MST algorithm becomes faster for non-pathological networks with better shortcut constructions.

Optimal Shortcuts for Planar Networks

Theorem. For any planar graph G=(V, E) and any partition into connected parts $S_1, S_2, \ldots, S_N$, there is a shortcut with O(D log D) congestion and O(D log D) dilation.

Remarks:

- There is a distributed algorithm computing these shortcuts in $\tilde{O}(D)$ rounds.
- Congestion + dilation = O(D log D) is existentially optimal, up to a log log D.

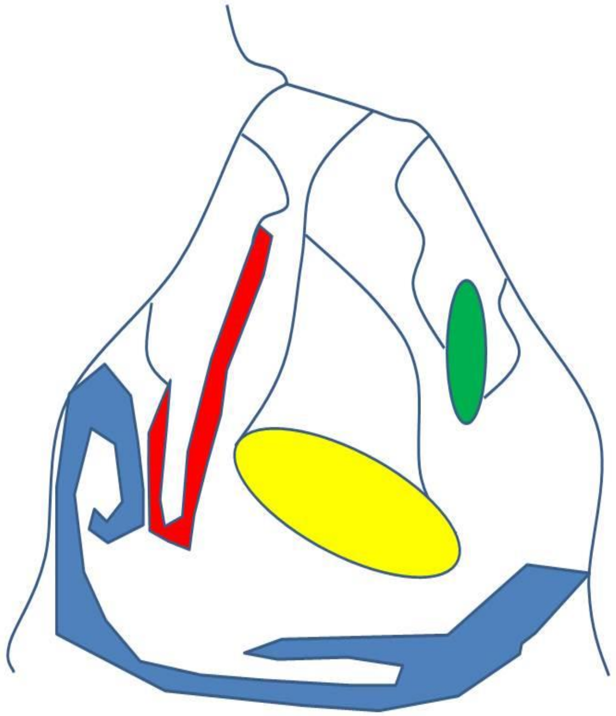
Corollary: A $\tilde{O}(D)$-round distributed algorithm for MST in planar networks.

Existence of Planar Shortcuts (simplified)

Theorem'. For any planar graph G=(V, E) and any partition, there is a shortcut with O(D) congestion and O(D^2) dilation.

Shortcut Definition:

- Fix a BFS-tree τ and a planar embedding of G
- Each part S_i has a left-most and right-most node l_i and r_i
- H_i is everything that is strictly enclosed by the cycle formed by the (l_i, r_i)-paths in S_i and the tree τ



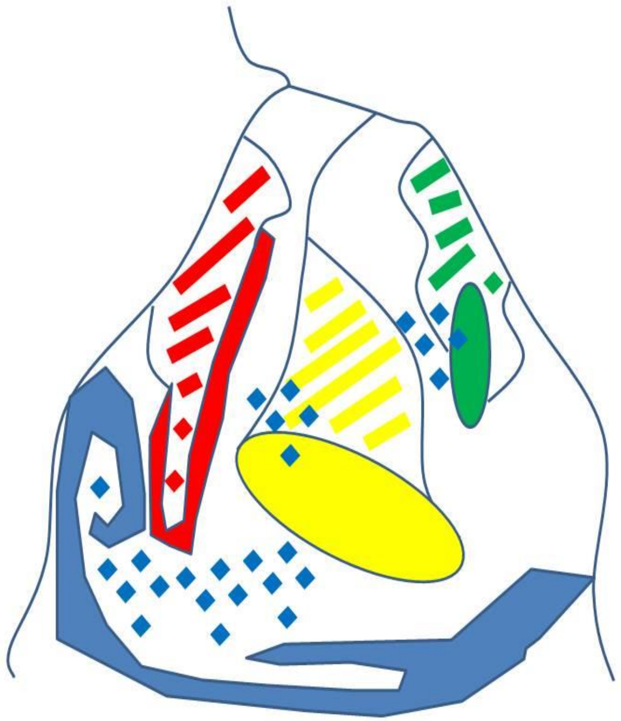
Existence of Planar Shortcuts (simplified)

Theorem'. For any planar graph G=(V, E) and any partition, there is a shortcut with O(D) congestion and O(D^2) dilation.

Shortcut Definition:

- Fix a BFS-tree τ and a planar embedding of G
- Each part S_i has a left-most and right-most node l_i and r_i
- H_i is everything that is strictly enclosed by the cycle formed by the (l_i, r_i)-paths in S_i and the tree τ

Congestion $\leq D$ because there are at most D sets *below* any edge.

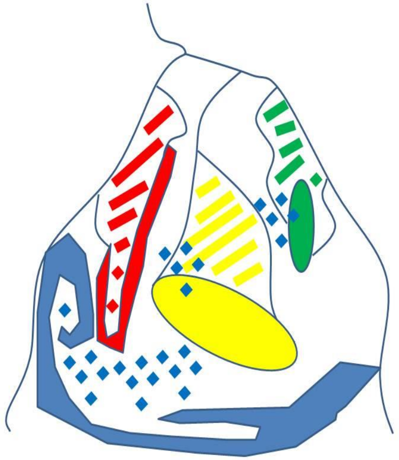


Existence of Planar Shortcuts (simplified)

Theorem'. For any planar graph G=(V, E) and any partition, there is a shortcut with O(D) congestion and O(D²) dilation.

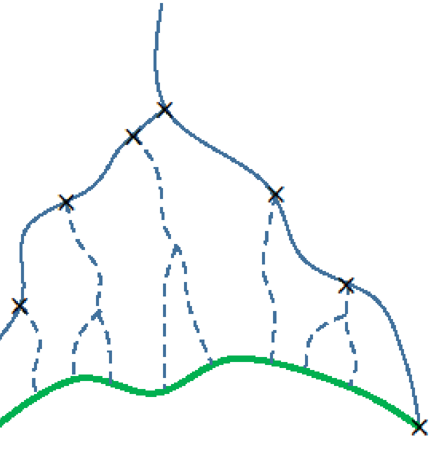
Shortcut Definition:

- Fix a BFS-tree τ and a planar embedding of G
- Each part S_i has a left-most and right-most node l_i and r_i
- $H_i =$ all tree edges that are strictly enclosed by the cycle formed by the (l_i, r_i)-paths in S_i and the tree τ



Congestion $\leq D$ because there are at most D sets *below* any edge.

Dilation D^2 routing: Shortcut S_i-path as far as possible within the enclosed subtree of τ, then go one step on the S_i-path. There are at most D subtrees of diameter D.



Further Results

Theorem [GH’16,HIZ’16b]:
Any network with **polylogarithmic genus**, **pathwidth, or treewidth** has shortcuts with $\tilde{O}(D)$ dilation and congestion for any partition.

All these shortcuts can be restricted to a shallow tree (e.g., a BFS-tree).

Theorem [HIZ’16a]:
There is a distributed algorithm which, for any network G, constructs a shortcut with congestion and dilation $\tilde{O}(Q)$ in $\tilde{O}(Q)$ rounds. Here Q is the quality of the best tree-restricted shortcut that exists in G.

We furthermore have efficient shortcut based distributed approximation algorithms for min-cut and many shortest-path type problems.

Putting everything together

We get distributed algorithms for

- MST
- $(1+\epsilon)$-approximate min-cut
- approximate shortest-path type problems

which run

- in $\tilde{O}(D)$ rounds on planar, bounded genus, or bounded treewidth networks
- in $\tilde{O}(\sqrt{n}+\ D)$ rounds on pathological worst-case networks

both of which are **instance optimal** (up to logarithmic factors).

The algorithms run generally about as fast as one can solve the partwise communication problem (using tree-restricted shortcuts) **in the given topology**.

Open Questions

- Solve further fundamental problems, e.g., Max-Flow, DFS, etc., with few invocations of Partwise Aggregation.
- Show formally that the Partwise Aggregation Problem is at least as hard as computing, e.g., an MST.
- Obtain an efficient distributed (polylog n)-approximation algorithm for computing quality low-congestion shortcuts.
- Further characterize network topologies with good shortcuts.
  (In particular, we believe that any minor-closed family of graphs has $\tilde{O}(D)$ quality tree-restricted shortcuts.)

Thanks!