Distributed Minimum Degree Spanning Trees

Michael Dinitz (Johns Hopkins University)
Magnús Halldórsson (Reykjavik University)
Taisuke Izumi (Nagoya Institute of Technology)
Calvin Newport (Georgetown University)

PODC 2019
August 2, 2019

Minimum Degree Spanning Tree

Input: Graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$
Feasible Solution: Spanning tree $\mathbf{T} = (\mathbf{V}, \mathbf{E}')$
Objective: Minimize max degree $\mathbf{\Delta}(\mathbf{T})$ of $\mathbf{T}$

Minimum Degree Spanning Tree

Input: Graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$
Feasible Solution: Spanning tree $\mathbf{T} = (\mathbf{V}, \mathbf{E'})$
Objective: Minimize max degree $\mathbf{\Delta(T)}$ of $\mathbf{T}$

- Classical NP-hard problem (reduction from Hamiltonian Path)
- Natural in distributed/networked settings:
  - Building a low-degree backbone network
  - Broadcast capacity in mobile telephone model [D, Halldórsson, Newport, Weaver DISC '19]
  - In many networking scenarios, degree ≈ load. Min max load.
  - Lots of attention to MST problem – why not MDST?

Algorithms for MDST

- Centralized ($\mathbf{d = OPT}$):
  - [Fürer, Raghavachari SODA '92]: Polytime $\mathbf{(d+1)}$-solution via complex recursive local search (*semi-local improvements*)
  - Simpler version (non-recursive) gives $\mathbf{(2d + \log n)}$-solution (*local improvements*)

Algorithms for MDST

- Centralized ($\mathbf{d = OPT}$):
  - [Fürer, Raghavachari SODA '92]: Polytime $\mathbf{(d+1)}$-solution via complex recursive local search (*semi-local improvements*)
  - Simpler version (non-recursive) gives $\mathbf{(2d + \log n)}$-solution (*local improvements*)
- Distributed:
  - [Blin, Butelle IPDPS '03]: Each local and semi-local improvement of FR can be computed in a distributed way
    - But separate improvements not computed in parallel, so still large running time ($\mathbf{\Omega(n)}$)
  - Self-stabilizing algorithms [Blin, Fraigniaud ICDCS '15], [Blin, Potop-Butucaru, Rovedakis '11]: running times $\mathbf{\Omega(n^2)}$
  - More general problem: find MST which minimizes max degree.
    - [Lavault, Valencia-Popon '08]: Still $\mathbf{\Omega(n^2)}$.

Algorithms for MDST

- Centralized ($\mathbf{d} = \mathbf{OPT}$):
  - [Fürer, Raghavachari SODA '92]: Polytime $\mathbf{(d+1)}$-solution via complex recursive local search (*semi-local improvements*)
  - Simpler version (non-recursive) gives $\mathbf{(2d+\log n)}$-solution (*local improvements*)
- Distributed:
  - [Blin, Butelle IPDPS '03]: Each local and semi-local improvement of FR can be computed in a distributed way
    - But separate improvements not computed in parallel, so still large running time $(\mathbf{\Omega(n)})$
  - Self-stabilizing algorithms [Blin, Fraigniaud ICDCS '15], [Blin, Potop-Butucaru, Rovedakis '11]: running times $\mathbf{\Omega(n^2)}$
  - More general problem: find MST which minimizes max degree.
    - [Lavault, Valencia-Popon '08]: Still $\mathbf{\Omega(n^2)}$.

Question: Can we provide good approximations for MDST in time $\mathbf{O(D+\sqrt{n})}$ (like for MST)?

Our Results: Upper Bounds

Models:

- CONGEST: synchronous rounds, each message size at most $\mathbf{O(\log n)}$ bits ($\mathbf{O(1)}$ words)
- broadcast-CONGEST: in each round, each node sends *same* message to all neighbors

Our Results: Upper Bounds

Models:

- CONGEST: synchronous rounds, each message size at most $\mathbf{O(\log n)}$ bits ($\mathbf{O(1)}$ words)
- broadcast-CONGEST: in each round, each node sends *same* message to all neighbors

Theorem

There is a randomized algorithm in the broadcast-CONGEST model which builds a spanning tree of maximum degree at most $\mathbf{4(1+\epsilon)d + O(\log n)}$ *and has expected running time at most* $\mathbf{O((D+\sqrt{n})\log^4 n)}$

Theorem

There is a deterministic algorithm in the CONGEST model which builds a spanning tree of maximum degree at most $\mathbf{4(1+\epsilon)d + O(\log n)}$ *and has expected running time at most* $\mathbf{O((D+\sqrt{n})\log^5 n)}$

Our Results: Lower Bounds

- Want to show that polynomial dependence on n is necessary
- Precise lower bound rather complex. Some simple corollaries:

Theorem

For any $\epsilon < 1/6$, there exists a family of instances of diameter $D = \Theta(n^{1/2-3\epsilon} + \log n)$ where any MDST algorithm with a polylogarithmic multiplicative approximation factor needs $\tilde{\Omega}(n^{1/2-\epsilon} + D)$ rounds.

Theorem

There exists a family of instances of diameter $D = O(\log n)$ where any deterministic MDST algorithm with a polylogarithmic multiplicative approximation factor needs $\tilde{\Omega}(n^{1/2})$ rounds.

Our Results: Lower Bounds

- Want to show that polynomial dependence on n is necessary
- Precise lower bound rather complex. Some simple corollaries:

Theorem

For any $\epsilon < 1/6$, there exists a family of instances of diameter $D = \Theta(n^{1/2-3\epsilon} + \log n)$ where any MDST algorithm with a polylogarithmic multiplicative approximation factor needs $\tilde{\Omega}(n^{1/2-\epsilon} + D)$ rounds.

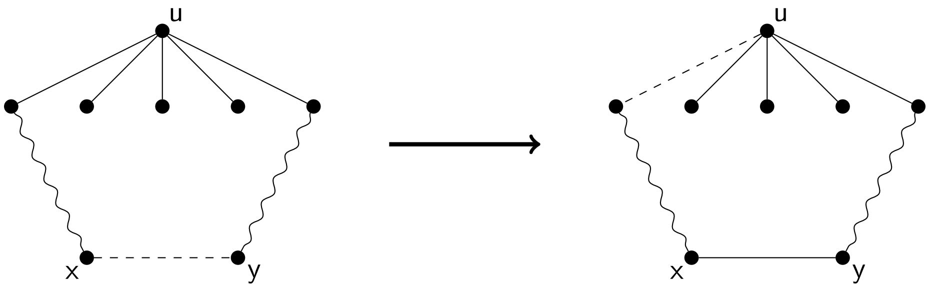
Theorem

There exists a family of instances of diameter $D = O(\log n)$ where any deterministic MDST algorithm with a polylogarithmic multiplicative approximation factor needs $\tilde{\Omega}(n^{1/2})$ rounds.

Today: only upper bounds

Local improvements

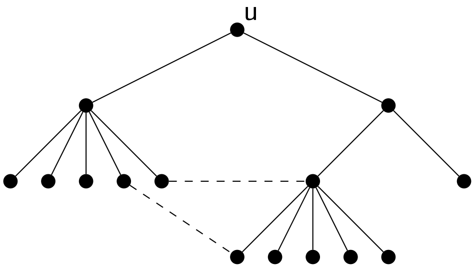
Idea: Adding an edge creates a cycle. Can remove any edge in that cycle.



- Degrees of $\mathbf{x}, \mathbf{y}$ increase by $\mathbf{1}$, but degree of $\mathbf{u}$ decreases by $\mathbf{1}$.
- FR: Make local improvements to decrease degrees of high-degree nodes, until no such local improvements exist.

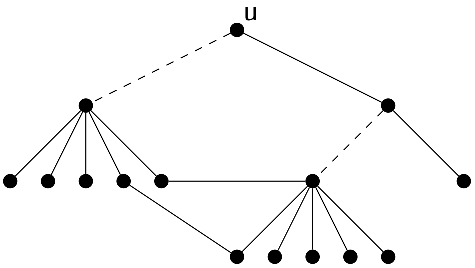
Parallel Improvements

To get $\mathbf{o(n)}$ running time, need to do many improvements in parallel. Requires coordination – not all local improvements can be done simultaneously!



Parallel Improvements

To get $\mathbf{o(n)}$ running time, need to do many improvements in parallel. Requires coordination – not all local improvements can be done simultaneously!



Everything else

- **Def:** Let $\mathbf{X_k}$ be nodes of degree at least $\mathbf{k}$ in current tree $\mathbf{T}$.
- Two parameters
  - γ: definition of "high-degree" (nodes in $\mathbf{X}_\gamma$)
  - $\mathbf{q}$: definition of "low-degree" (degree at most $\gamma_0 := \gamma - 2\mathbf{q}$)
- Local improvements:
  - Hurt low-degree nodes by at most $\mathbf{q}$
  - Improve high-degree nodes, but by at most $\mathbf{q}$ (so still worse than low-degree nodes)

Everything else

- **Def:** Let $\mathbf{X_k}$ be nodes of degree at least $\mathbf{k}$ in current tree $\mathbf{T}$.
- Two parameters
  - γ: definition of "high-degree" (nodes in $\mathbf{X}_\gamma$)
  - $\mathbf{q}$: definition of "low-degree" (degree at most $\gamma_0 := \gamma - 2\mathbf{q}$)
- Local improvements:
  - Hurt low-degree nodes by at most $\mathbf{q}$
  - Improve high-degree nodes, but by at most $\mathbf{q}$ (so still worse than low-degree nodes)

Informal Theorem: If (some conditions), then we can find $\Omega(|\mathbf{X}_\gamma \mathbf{q}|)$ local improvements in $\tilde{O}(D + \sqrt{n})$ rounds that can all be done simultaneously.

Everything else

- **Def:** Let $\mathbf{X_k}$ be nodes of degree at least $\mathbf{k}$ in current tree $\mathbf{T}$.
- Two parameters
  - γ: definition of "high-degree" (nodes in $\mathbf{X}_\gamma$)
  - $\mathbf{q}$: definition of "low-degree" (degree at most $\gamma_0 := \gamma - 2\mathbf{q}$)
- Local improvements:
  - Hurt low-degree nodes by at most $\mathbf{q}$
  - Improve high-degree nodes, but by at most $\mathbf{q}$ (so still worse than low-degree nodes)

Informal Theorem: If (some conditions), then we can find $\Omega(|\mathbf{X}_\gamma \mathbf{q}|)$ local improvements in $\tilde{O}(D + \sqrt{n})$ rounds that can all be done simultaneously.

- Fix $\mathbf{q}$, find γ which gives "large" improvement. Repeat until no such γ.
- Try all $\mathbf{q}$
- Complicated potential function to prove $\tilde{O}(1)$ iterations
- Approximation basically from centralized analysis (some extra loss)

Everything else

- **Def:** Let $\mathbf{X_k}$ be nodes of degree at least $\mathbf{k}$ in current tree $\mathbf{T}$.
- Two parameters
  - γ: definition of "high-degree" (nodes in $\mathbf{X}_\gamma$)
  - $\mathbf{q}$: definition of "low-degree" (degree at most $\gamma_0 := \gamma - 2\mathbf{q}$)
- Local improvements:
  - Hurt low-degree nodes by at most $\mathbf{q}$
  - Improve high-degree nodes, but by at most $\mathbf{q}$ (so still worse than low-degree nodes)

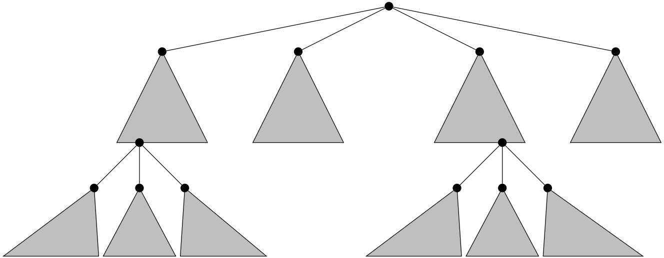
Informal Theorem: If (some conditions), then we can find $\Omega(|\mathbf{X}_\gamma \mathbf{q}|)$ local improvements in $\tilde{O}(D + \sqrt{n})$ rounds that can all be done simultaneously.

- Fix $\mathbf{q}$, find γ which gives "large" improvement. Repeat until no such γ.
- Try all $\mathbf{q}$
- Complicated potential function to prove $\tilde{O}(1)$ iterations
- Approximation basically from centralized analysis (some extra loss)

Main point (rest of talk): want to find lots of parallel local improvements.

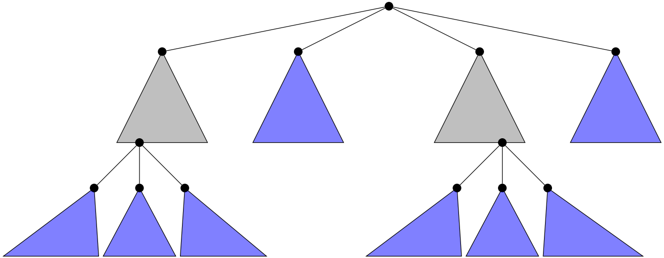
Leaf Branches

- *Branch*: Component of $\mathbf{T} \setminus \mathbf{X}_\gamma$
- *Leaf branch*: branch $\mathbf{C}$ where there is only one edge in $\mathbf{T}$ leaving $\mathbf{C}$.



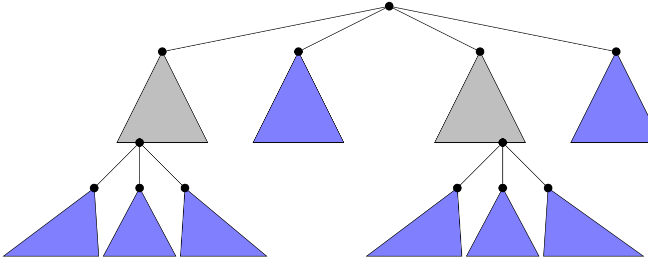
Leaf Branches

- *Branch*: Component of $\mathbf{T} \setminus \mathbf{X}_\gamma$
- *Leaf branch*: branch $\mathbf{C}$ where there is only one edge in $\mathbf{T}$ leaving $\mathbf{C}$.



Leaf Branches

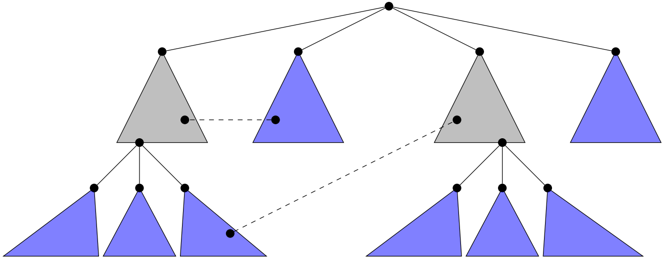
- *Branch*: Component of $\mathbf{T} \setminus \mathbf{X}_\gamma$
- *Leaf branch*: branch **C** where there is only one edge in **T** leaving **C**.



If add edges between leaf and non-leaf branches, parallel improvements don't interfere!

Leaf Branches

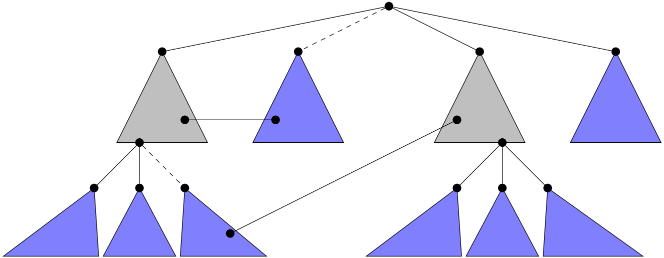
- *Branch*: Component of $\mathbf{T} \setminus \mathbf{X}_\gamma$
- *Leaf branch*: branch **C** where there is only one edge in **T** leaving **C**.



If add edges between leaf and non-leaf branches, parallel improvements don't interfere!

Leaf Branches

- *Branch*: Component of $\mathbf{T} \setminus \mathbf{X}_\gamma$
- *Leaf branch*: branch **C** where there is only one edge in **T** leaving **C**.



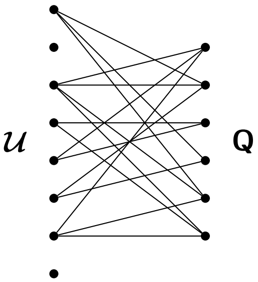
If add edges between leaf and non-leaf branches, parallel improvements don't interfere!

Improvement Graph

- Edge is *good* if:
  - Both endpoints degree at most γ_0
  - Endpoints in different branches, one of which is a leaf branch
- We'll only make improvements from adding good edges.

Improvement Graph

- Edge is *good* if:
  - Both endpoints degree at most γ_0
  - Endpoints in different branches, one of which is a leaf branch
- We'll only make improvements from adding good edges.
- Create new bipartite graph: *improvement graph*

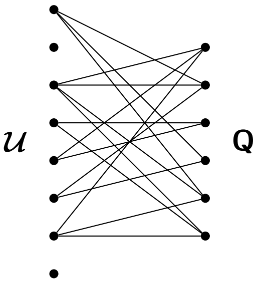


- $\mathcal{U}$: leaf branches
- $\mathbf{Q}$: nodes of degree $\leq \gamma_0$
- Edges: good edges in $\mathbf{G}$

Approach: Find a large $(\mathbf{1}, \mathbf{q})$-matching in improvement graph (degree $\mathbf{1}$ on leaf branches, degree $\leq \mathbf{q}$ on low-degree nodes).

Improvement Graph

- Edge is *good* if:
  - Both endpoints degree at most γ_0
  - Endpoints in different branches, one of which is a leaf branch
- We'll only make improvements from adding good edges.
- Create new bipartite graph: *improvement graph*



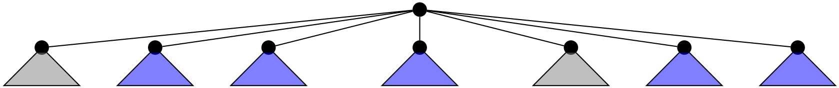
- $\mathcal{U}$: leaf branches
- $\mathbf{Q}$: nodes of degree $\le \gamma_0$
- Edges: good edges in $\mathbf{G}$

Approach: Find a large $(\mathbf{1}, \mathbf{q})$-matching in improvement graph (degree $\mathbf{1}$ on leaf branches, degree $\le \mathbf{q}$ on low-degree nodes).

- What about requirement that high-degree edges improve at most $\mathbf{q}$?

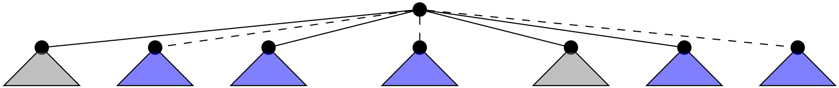
Constrained **q**-Matching

High-degree node improves ≥ **q** only if ≥ **q** leaf branches adjacent to it are matched.



Constrained **q**-Matching

High-degree node improves ≥ **q** only if ≥ **q** leaf branches adjacent to it are matched.

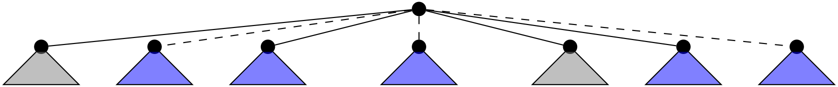


Constrained **q**-*Matching*: force this not to happen

- Partition $\mathcal{U}$ into *bundles* which share a parent
- Require each bundle to only have **q** matched nodes.

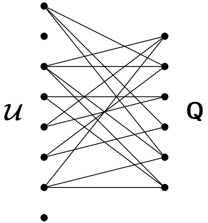
Constrained **q**-Matching

High-degree node improves $\geq \mathbf{q}$ only if $\geq \mathbf{q}$ leaf branches adjacent to it are matched.

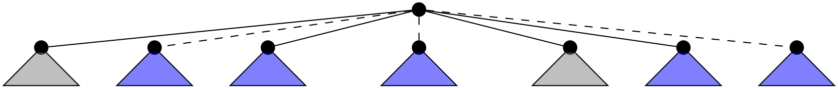


Constrained **q**-*Matching*: force this not to happen

- Partition $\mathcal{U}$ into *bundles* which share a parent
- Require each bundle to only have **q** matched nodes.

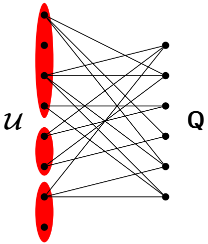


High-degree node improves ≥ **q** only if ≥ **q** leaf branches adjacent to it are matched.



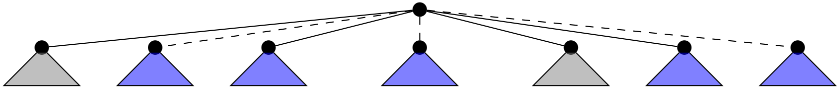
Constrained **q**-*Matching*: force this not to happen

- Partition $\mathcal{U}$ into *bundles* which share a parent
- Require each bundle to only have **q** matched nodes.



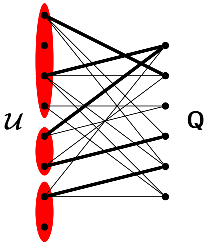
Constrained **q**-Matching

High-degree node improves $\geq \mathbf{q}$ only if $\geq \mathbf{q}$ leaf branches adjacent to it are matched.



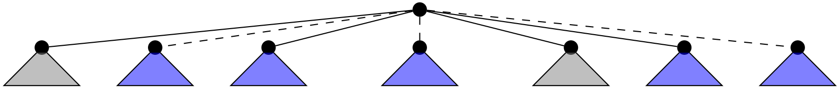
Constrained **q**-*Matching*: force this not to happen

- Partition $\mathcal{U}$ into *bundles* which share a parent
- Require each bundle to only have **q** matched nodes.



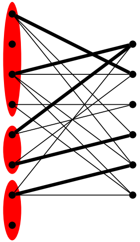
Constrained **q**-Matching

High-degree node improves $\geq \mathbf{q}$ only if $\geq \mathbf{q}$ leaf branches adjacent to it are matched.



Constrained **q**-*Matching*: force this not to happen

- Partition $\mathcal{U}$ into *bundles* which share a parent
- Require each bundle to only have **q** matched nodes.



So want to prove:

- There must exist a large Constrained **q**-Matching (if γ large enough and not many "medium-degree" nodes)
- Can quickly find $\mathbf{O(1)}$-approximation to Max Constrained **q**-Matching.

Large Constrained $\mathbf{q}$-Matching

Theorem

There is a constrained $\mathbf{q}$*-matching of size at least* $\frac{\mathbf{q}}{\gamma}((\gamma - \mathbf{2})|\mathbf{X}_\gamma| - \mathbf{d}|\mathbf{X}_{\gamma_0}|)$ *(where* $\mathbf{d}$ *is optimal max degree)*

Large Constrained $\mathbf{q}$-Matching

Theorem

There is a constrained $\mathbf{q}$*-matching of size at least* $\frac{\mathbf{q}}{\gamma}((\gamma-2)|\mathbf{X}_\gamma| - \mathbf{d}|\mathbf{X}_{\gamma_0}|)$ *(where* $\mathbf{d}$ *is optimal max degree)*

Proof Sketch:

- Each node in $\mathbf{X}_\gamma$ causes lots of leaf branches when removed.
- **OPT** connects these components without using many edges.
- Counting and averaging.

Large Constrained $\mathbf{q}$-Matching

Theorem

There is a constrained $\mathbf{q}$*-matching of size at least* $\frac{\mathbf{q}}{\gamma}\left((\gamma-\mathbf{2})|\mathbf{X}_\gamma| - \mathbf{d}|\mathbf{X}_{\gamma_0}|\right)$ *(where* $\mathbf{d}$ *is optimal max degree)*

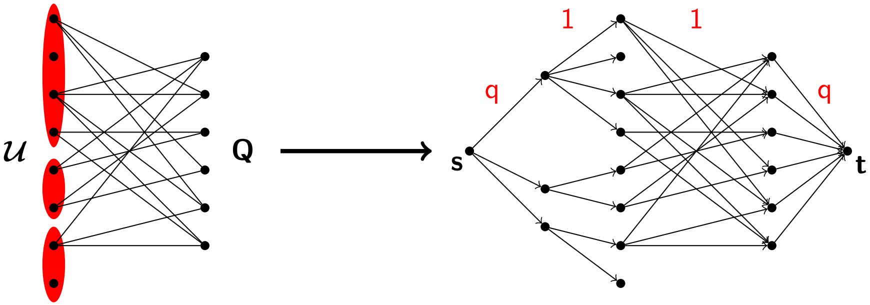
Proof Sketch:

- Each node in $\mathbf{X}_\gamma$ causes lots of leaf branches when removed.
- **OPT** connects these components without using many edges.
- Counting and averaging.

So if $|\mathbf{X}_{\gamma_0}|$ not much larger than $\mathbf{X}_\gamma$ and γ large enough, about $\mathbf{q}|\mathbf{X}_\gamma|$

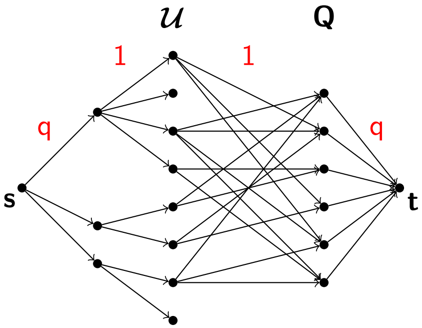
Approximating Constrained **q**-Matching

Turn constrained **q**-matching into flow:



- Integral capacities, so integral max flow
- Flow of α iff constrained **q**-matching of size α

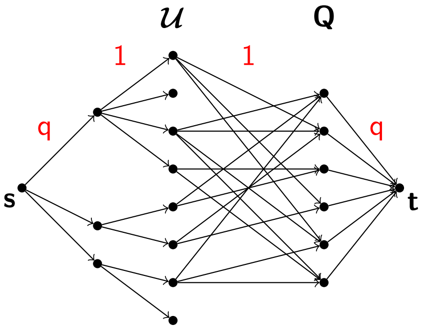
Approximate Max Flow



Lemma: In depth-**d** flow network, any maximal flow has value at least **(1/d)** of max flow.

- High-Level Algorithm:
  - Start each flow path with **1/m** flow
  - In each round: double flow in each flow path, unless some node on path already "full"

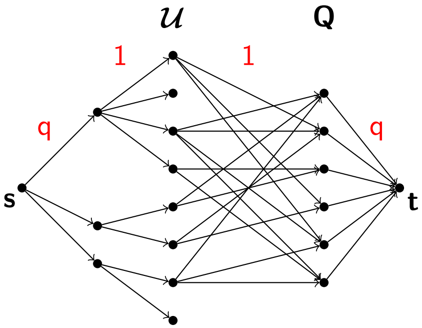
Approximate Max Flow



Lemma: In depth-**d** flow network, any maximal flow has value at least **(1/d)** of max flow.

- High-Level Algorithm:
  - Start each flow path with **1/m** flow
  - In each round: double flow in each flow path, unless some node on path already "full"
- Gives approximate fractional maximal flow.
- Randomized rounding to get approximate constrained **q**-matching

Approximate Max Flow



Lemma: In depth-**d** flow network, any maximal flow has value at least **(1/d)** of max flow.

- High-Level Algorithm:
  - Start each flow path with **1/m** flow
  - In each round: double flow in each flow path, unless some node on path already "full"
- Gives approximate fractional maximal flow.
- Randomized rounding to get approximate constrained **q**-matching
- Some annoying details ($\mathcal{U}$ are actually components, not nodes)

Putting it Together

- Want to find many parallel local improvements
- Restrict to "safe" improvements: add good edge, remove edge from leaf branch
- Prove there have to be many safe improvements (or else finished)
- Algorithm for approximating max safe improvements via max flow, randomized rounding.
- $\tilde{\mathbf{O}}(\mathbf{D} + \sqrt{\mathbf{n}})$ rounds using complex but standard CONGEST communication ideas.

Putting it Together

- Want to find many parallel local improvements
- Restrict to "safe" improvements: add good edge, remove edge from leaf branch
- Prove there have to be many safe improvements (or else finished)
- Algorithm for approximating max safe improvements via max flow, randomized rounding.
- $\tilde{\mathbf{O}}(\mathbf{D} + \sqrt{\mathbf{n}})$ rounds using complex but standard CONGEST communication ideas.

- Have to do this for polylog values of $\mathbf{q}$.
- For each $\mathbf{q}$, make progress on potential function or finish.

Putting it Together

- Want to find many parallel local improvements
- Restrict to "safe" improvements: add good edge, remove edge from leaf branch
- Prove there have to be many safe improvements (or else finished)
- Algorithm for approximating max safe improvements via max flow, randomized rounding.
- $\tilde{O}(D + \sqrt{n})$ rounds using complex but standard CONGEST communication ideas.

- Have to do this for polylog values of q.
- For each q, make progress on potential function or finish.

- Approximation guarantee:
  - Somewhat complex/delicate
  - Morally: same as centralized FR, but with small extra loss (2 to 4 multiplicative)

Conclusion

Results:

- Gave first $\tilde{\mathbf{O}}(\mathbf{D}+\sqrt{\mathbf{n}})$-round $\mathbf{O}(\mathbf{d}+\log \mathbf{n})$-degree bounded MDST algorithm in CONGEST
- Lower bound: Can't really improve running time even if only want $\mathbf{O}(\mathbf{d}\cdot\log \mathbf{n})$-degree

Conclusion

Results:

- Gave first $\tilde{O}(D + \sqrt{n})$-round $O(d + \log n)$-degree bounded MDST algorithm in CONGEST
- Lower bound: Can't really improve running time even if only want $O(d \cdot \log n)$-degree

Open Problems:

- Key idea was finding many local improvements which can be done in parallel. Should imply PRAM, streaming, etc. algorithms?
- Stronger approximation! Ideally: degree $d + 1$?
- Steiner tree instead of spanning?
  - Centralized bounds same (degree $d + 1$ from local search)
  - Centralized algorithm makes local improvements using *paths* instead of edges. Distributed?

Thanks!