Tree-Packing Revisited: Faster Fully Dynamic Min-Cut and Arboricity

SODA 2025

Tijn de Vos (r) Aleksander B.G. Christiansen

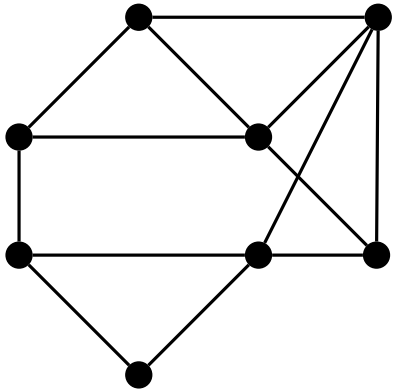
TU Graz

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 947702) and is supported by the Austrian Science Fund (FWF): P 32863-N.

Der Wissenschaftsfonds.

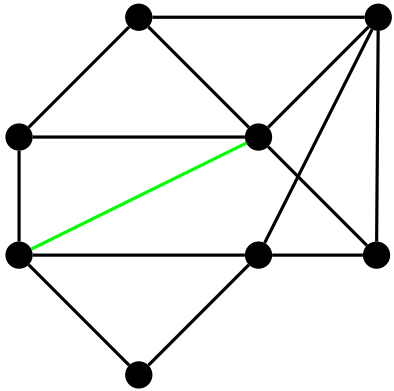
Dynamic Graph Algorithms

- Graph $G = (V, E)$ with edge updates: insertions/deletions.



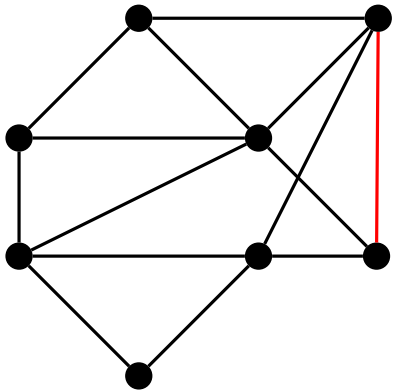
Dynamic Graph Algorithms

- Graph $G = (V, E)$ with edge updates: insertions/deletions.



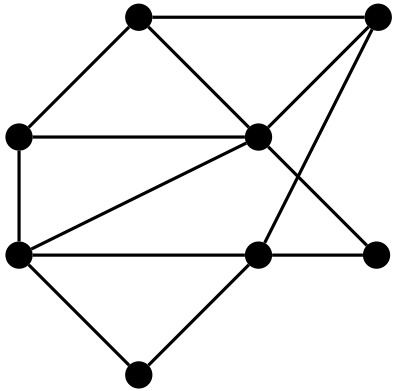
Dynamic Graph Algorithms

- Graph $G = (V, E)$ with edge updates: insertions/deletions.



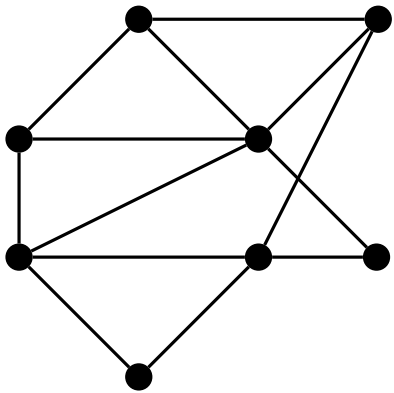
Dynamic Graph Algorithms

- Graph $G = (V, E)$ with edge updates: insertions/deletions.



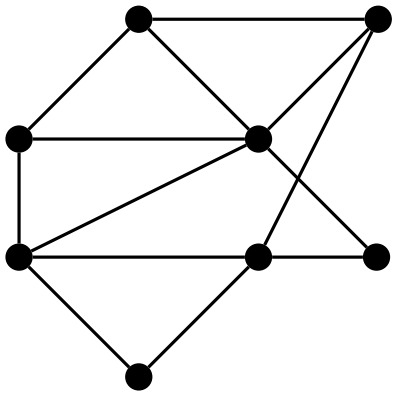
Dynamic Graph Algorithms

- Graph $G = (V, E)$ with edge updates: insertions/deletions.
- Maintain the answer to a problem.



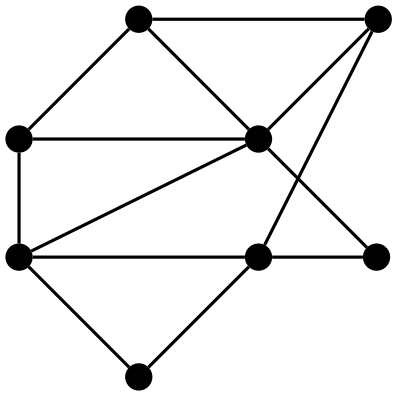
Dynamic Graph Algorithms

- Graph $G = (V, E)$ with edge updates: insertions/deletions.
- Maintain the answer to a problem.
- Goal: low update time.
  - Worst-case.
  - Amortized = 'on average'.



Dynamic Graph Algorithms

- Graph $G = (V, E)$ with edge updates: insertions/deletions.
- Maintain the answer to a problem.
- Goal: low update time.
  - Worst-case.
  - Amortized = 'on average'.
- Adversary determines updates:
  - Oblivious Adversary: updates fixed beforehand.
  - Adaptive Adversary: updates can depend on (random) choices of the algorithm.



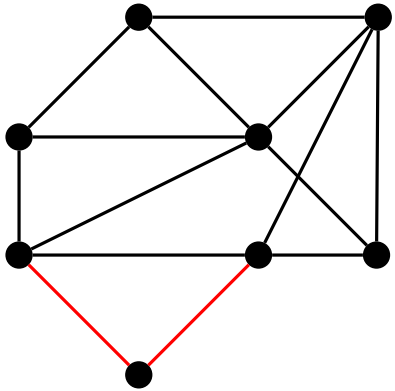
Part 1: Dynamic Min-Cut

Part 2: Dynamic Arboricity

Part 3: More on Tree-Packings

Problem

- Min-Cut (connectivity): find λ, the minimum number of edges disconnecting the graph.



Dynamic Min-Cut: Main Result

Theorem

Unweighted, undirected (multi-)graph $G = (V, E)$. There exists a deterministic dynamic algorithm that maintains the exact min-cut value λ if $\lambda \leq \lambda_{\max}$ in $\tilde{O}(\lambda_{\max}^{5.5}\sqrt{n})$ worst-case update time.

Previously: $\tilde{O}(\lambda_{\max}^{14.5}\sqrt{n})$ [Thorup '07].

Dynamic Min-Cut: Main Result

Theorem

Unweighted, undirected (multi-)graph $G = (V, E)$. There exists a deterministic dynamic algorithm that maintains the exact min-cut value λ if $\lambda \leq \lambda_{\max}$ in $\tilde{O}(\lambda_{\max}^{5.5}\sqrt{n})$ worst-case update time.

Previously: $\tilde{O}(\lambda_{\max}^{14.5}\sqrt{n})$ [Thorup '07].

Corollary

Simple, unweighted, undirected graph $G = (V, E)$. There exists a deterministic dynamic algorithm that maintains the exact min-cut value λ with amortized update time

$$\tilde{O}(\min\{m^{1-1/12}, m^{11/13}n^{1/13}, n^{3/2}\}).$$

Previously: $\tilde{O}(m^{1-1/31})$ [Goranci/Henzinger/Nanongkai/Saranurak/Thorup/Wulff-Nilsen '23].

Tree-Packing and Min-Cut

Definitions

- Tree-packing $\mathcal{T} = \{T_1, T_2, \dots\}$: a family of trees in G such that [...].
- Load $L^{\mathcal{T}}(e)$: the number of trees in $\mathcal{T}$ that contain an edge e.
- Relative load $\ell^{\mathcal{T}}(e) := L^{\mathcal{T}}(e)/|\mathcal{T}|$.

Tree-Packing and Min-Cut

Definitions

- Tree-packing $\mathcal{T} = \{T_1, T_2, \dots\}$: a family of trees in G such that [...].
- Load $L^{\mathcal{T}}(e)$: the number of trees in $\mathcal{T}$ that contain an edge e.
- Relative load $\ell^{\mathcal{T}}(e) := L^{\mathcal{T}}(e)/|\mathcal{T}|$.

- Static [Karger '00] + [...].
- Distributed [Daga/Henzinger/Nanongkai/Saranurak '19, Dory/Efron/Mukhopadhyay/Nanongkai '21].
- Dynamic [Thorup/Karger '00, Thorup '07].

Static: Disjoint Trees [Karger '00]

Theorem [Nash-Williams '61, Tutte '61]

A graph with minimum cut λ contains at least $\lambda/2$ disjoint spanning trees.

- Take $\mathcal{T} = \{T_1, T_2, \dots\}$ a maximum packing of disjoint spanning trees.

Static: Disjoint Trees [Karger '00]

Theorem [Nash-Williams '61, Tutte '61]

A graph with minimum cut λ contains at least $\lambda/2$ disjoint spanning trees.

- Take $\mathcal{T} = \{T_1, T_2, \dots\}$ a maximum packing of disjoint spanning trees.
- We know $\lambda/2 \leq |\mathcal{T}| \leq \lambda$ by above.

Static: Disjoint Trees [Karger '00]

Theorem [Nash-Williams '61, Tutte '61]

A graph with minimum cut λ contains at least $\lambda/2$ disjoint spanning trees.

- Take $\mathcal{T} = \{T_1, T_2, \dots\}$ a maximum packing of disjoint spanning trees.
- We know $\lambda/2 \leq |\mathcal{T}| \leq \lambda$ by above.
- $\mathcal{T}$ covers any min-cut C. On average, a tree contains $\lambda/|\mathcal{T}| \leq \lambda/(\lambda/2) \leq 2$ edges from C: it 2-respects C.

Static: Disjoint Trees [Karger '00]

Theorem [Nash-Williams '61, Tutte '61]

A graph with minimum cut λ contains at least $\lambda/2$ disjoint spanning trees.

- Take $\mathcal{T} = \{T_1, T_2, \dots\}$ a maximum packing of disjoint spanning trees.
- We know $\lambda/2 \leq |\mathcal{T}| \leq \lambda$ by above.
- $\mathcal{T}$ covers any min-cut C. On average, a tree contains $\lambda/|\mathcal{T}| \leq \lambda/(\lambda/2) \leq 2$ edges from C: it 2-respects C.
- Can find tree-packing of size $O(\lambda)$ to approximate a packing of disjoint trees [Gabow '95, Plotkin/Shmoys/Tardos '91].

Static: Disjoint Trees [Karger '00]

Theorem [Nash-Williams '61, Tutte '61]

A graph with minimum cut λ contains at least $\lambda/2$ disjoint spanning trees.

- Take $\mathcal{T} = \{T_1, T_2, \dots\}$ a maximum packing of disjoint spanning trees.
- We know $\lambda/2 \leq |\mathcal{T}| \leq \lambda$ by above.
- $\mathcal{T}$ covers any min-cut C. On average, a tree contains $\lambda/|\mathcal{T}| \leq \lambda/(\lambda/2) \leq 2$ edges from C: it 2-respects C.
- Can find tree-packing of size $O(\lambda)$ to approximate a packing of disjoint trees [Gabow '95, Plotkin/Shmoys/Tardos '91].
- Can find 2-respecting cuts in $O(m \log^2 n)$ time.

Static: Disjoint Trees [Karger '00]

Theorem [Nash-Williams '61, Tutte '61]

A graph with minimum cut λ contains at least $\lambda/2$ disjoint spanning trees.

- Take $\mathcal{T} = \{T_1, T_2, \dots\}$ a maximum packing of disjoint spanning trees.
- We know $\lambda/2 \leq |\mathcal{T}| \leq \lambda$ by above.
- $\mathcal{T}$ covers any min-cut C. On average, a tree contains $\lambda/|\mathcal{T}| \leq \lambda/(\lambda/2) \leq 2$ edges from C: it 2-respects C.
- Can find tree-packing of size $O(\lambda)$ to approximate a packing of disjoint trees [Gabow '95, Plotkin/Shmoys/Tardos '91].
- Can find 2-respecting cuts in $O(m \log^2 n)$ time.
- Sample $\lambda \to \log n$ to obtain $O(m \log^3 n)$ time.

Ideal Tree-Packing

- Define ideal load $\ell^*(e)$.

Ideal Tree-Packing

- Define ideal load $\ell^*(e)$.
- We want

$$\#\text{disjoint spanning trees} = \frac{1}{\max_e \ell^*(e)}.$$

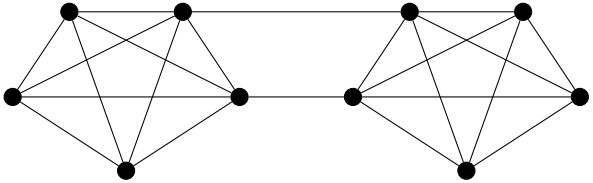
Ideal Tree-Packing

- Define ideal load $\ell^*(e)$.
- We want

$$\#\text{disjoint spanning trees} = \frac{1}{\max_e \ell^*(e)}.$$

- If we fit k disjoint spanning trees in some $G/\mathcal{P}$ containing e then

$$\ell^*(e) = \frac{1}{k}.$$



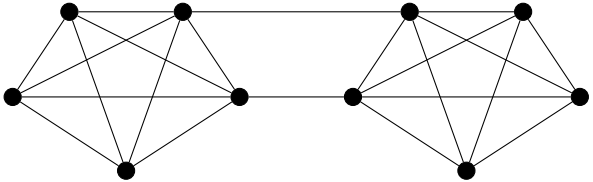
Ideal Tree-Packing

- Define ideal load $\ell^*(e)$.
- We want

$$\#\text{disjoint spanning trees} = \frac{1}{\max_e \ell^*(e)}.$$

- If we fit k disjoint spanning trees in some $G/\mathcal{P}$ containing e then

$$\ell^*(e) = \frac{1}{k}.$$



- Spoiler:

$$\text{arboricity} = \frac{1}{\min_{e \in E} \ell^*(e)}.$$

Ideal Tree-Packing – Formal

- For a vertex partition $\mathcal{P}$, we define the partition value as

$$\text{part\_val}(\mathcal{P}) := \frac{|\mathrm{E}(\mathrm{G}/\mathcal{P})|}{|\mathcal{P}| - 1}.$$

Ideal Tree-Packing – Formal

- For a vertex partition $\mathcal{P}$, we define the partition value as

$$\text{part\_val}(\mathcal{P}) := \frac{|\mathrm{E}(\mathrm{G}/\mathcal{P})|}{|\mathcal{P}| - 1}.$$

- Rephrase [Nash-Williams '61, Tutte '61]:

$$\Phi := \max_{\mathcal{T}} \frac{1}{\max_{e \in E} \ell^{\mathcal{T}}(e)} = \min_{\mathcal{P}} \text{part\_val}(\mathcal{P}).$$

Ideal Tree-Packing – Formal

- For a vertex partition $\mathcal{P}$, we define the partition value as

$$\text{part\_val}(\mathcal{P}) := \frac{|\mathrm{E}(\mathrm{G}/\mathcal{P})|}{|\mathcal{P}| - 1}.$$

- Rephrase [Nash-Williams '61, Tutte '61]:

$$\Phi := \max_{\mathcal{T}} \frac{1}{\max_{e \in E} \ell^{\mathcal{T}}(e)} = \min_{\mathcal{P}} \text{part\_val}(\mathcal{P}).$$

- Define ideal load ℓ^* recursively:

Ideal Tree-Packing – Formal

- For a vertex partition $\mathcal{P}$, we define the partition value as

$$\text{part\_val}(\mathcal{P}) := \frac{|\mathrm{E}(\mathrm{G}/\mathcal{P})|}{|\mathcal{P}| - 1}.$$

- Rephrase [Nash-Williams '61, Tutte '61]:

$$\Phi := \max_{\mathcal{T}} \frac{1}{\max_{e \in E} \ell^{\mathcal{T}}(e)} = \min_{\mathcal{P}} \text{part\_val}(\mathcal{P}).$$

- Define ideal load ℓ^* recursively:
  - Let $\mathcal{P}^*$ be a partition with $\text{part\_val}(\mathcal{P}^*) = \Phi$.

Ideal Tree-Packing – Formal

- For a vertex partition $\mathcal{P}$, we define the partition value as

$$\mathrm{part\_val}(\mathcal{P}) := \frac{|\mathrm{E}(\mathrm{G}/\mathcal{P})|}{|\mathcal{P}| - 1}.$$

- Rephrase [Nash-Williams '61, Tutte '61]:

$$\Phi := \max_{\mathcal{T}} \frac{1}{\max_{e \in E} \ell^{\mathcal{T}}(e)} = \min_{\mathcal{P}} \mathrm{part\_val}(\mathcal{P}).$$

- Define ideal load ℓ^* recursively:
  - Let $\mathcal{P}^*$ be a partition with $\mathrm{part\_val}(\mathcal{P}^*) = \Phi$.
  - For all $e \in E(G/\mathcal{P}^*)$, set $\ell^*(e) := 1/\Phi$.

Ideal Tree-Packing – Formal

- For a vertex partition $\mathcal{P}$, we define the partition value as

$$\mathrm{part\_val}(\mathcal{P}) := \frac{|\mathrm{E}(\mathrm{G}/\mathcal{P})|}{|\mathcal{P}| - 1}.$$

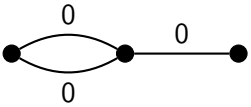
- Rephrase [Nash-Williams '61, Tutte '61]:

$$\Phi := \max_{\mathcal{T}} \frac{1}{\max_{e \in E} \ell^{\mathcal{T}}(e)} = \min_{\mathcal{P}} \mathrm{part\_val}(\mathcal{P}).$$

- Define ideal load ℓ^* recursively:
  - Let $\mathcal{P}^*$ be a partition with $\mathrm{part\_val}(\mathcal{P}^*) = \Phi$.
  - For all $e \in E(G/\mathcal{P}^*)$, set $\ell^*(e) := 1/\Phi$.
  - For each $S \in \mathcal{P}^*$, recurse on the subgraph $G[S]$.

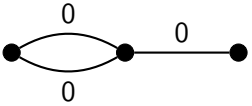
Greedy Tree-Packing

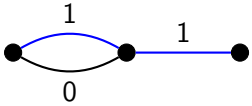
- Let the weight of an edge be the number of trees an edge belongs to. In a greedy tree-packing $\mathcal{T}$, the spanning trees form successive minimum spanning trees.



Greedy Tree-Packing

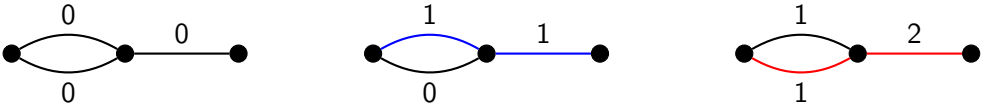
- Let the weight of an edge be the number of trees an edge belongs to. In a greedy tree-packing $\mathcal{T}$, the spanning trees form successive minimum spanning trees.





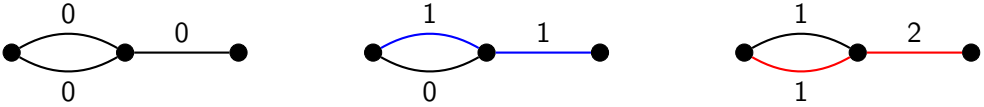
Greedy Tree-Packing

- Let the weight of an edge be the number of trees an edge belongs to. In a greedy tree-packing $\mathcal{T}$, the spanning trees form successive minimum spanning trees.



Greedy Tree-Packing

- Let the weight of an edge be the number of trees an edge belongs to. In a greedy tree-packing $\mathcal{T}$, the spanning trees form successive minimum spanning trees.



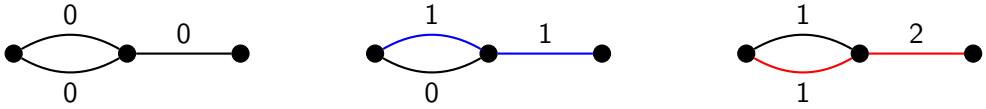
- If $|\mathcal{T}| \geq 6\lambda \log m/\varepsilon^2$ then

$$|\ell^{\mathcal{T}}(e) - \ell^*(e)| \leq \varepsilon/\lambda,$$

for all $e \in E$ [Thorup '07].

Greedy Tree-Packing

- Let the weight of an edge be the number of trees an edge belongs to. In a greedy tree-packing $\mathcal{T}$, the spanning trees form successive minimum spanning trees.



- If $|\mathcal{T}| \geq 6\lambda \log m/\varepsilon^2$ then

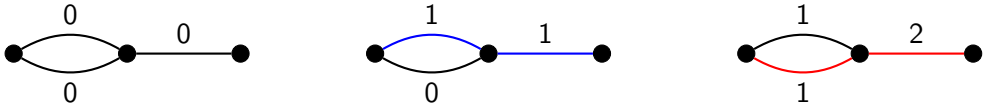
$$|\ell^{\mathcal{T}}(e) - \ell^*(e)| \leq \varepsilon/\lambda,$$

for all $e \in E$ [Thorup '07].

  - If $|\mathcal{T}| \geq \Theta(\lambda \log n)$, then at least one tree 2-respects the min-cut.

Greedy Tree-Packing

- Let the weight of an edge be the number of trees an edge belongs to. In a greedy tree-packing $\mathcal{T}$, the spanning trees form successive minimum spanning trees.



- If $|\mathcal{T}| \geq 6\lambda \log m/\varepsilon^2$ then

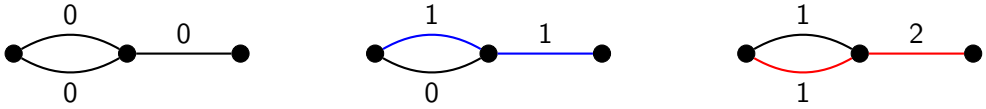
$$|\ell^{\mathcal{T}}(e) - \ell^*(e)| \leq \varepsilon/\lambda,$$

for all $e \in E$ [Thorup '07].

  - If $|\mathcal{T}| \geq \Theta(\lambda \log n)$, then at least one tree 2-respects the min-cut.
  - If $|\mathcal{T}| \geq \Theta(\lambda^7 \log n)$ then at least one tree 1-respects the min-cut. [Arkhipov/Kolmogorov '25]: $|\mathcal{T}| \geq \Theta(\lambda^5 \log n)$.

Greedy Tree-Packing

- Let the weight of an edge be the number of trees an edge belongs to. In a greedy tree-packing $\mathcal{T}$, the spanning trees form successive minimum spanning trees.



- If $|\mathcal{T}| \geq 6\lambda \log m/\varepsilon^2$ then

$$|\ell^{\mathcal{T}}(e) - \ell^*(e)| \leq \varepsilon/\lambda,$$

for all $e \in E$ [Thorup '07].

  - If $|\mathcal{T}| \geq \Theta(\lambda \log n)$, then at least one tree 2-respects the min-cut.
  - If $|\mathcal{T}| \geq \Theta(\lambda^7 \log n)$ then at least one tree 1-respects the min-cut. [Arkhipov/Kolmogorov '25]: $|\mathcal{T}| \geq \Theta(\lambda^5 \log n)$.

- Distributed [Daga/Henzinger/Nanongkai/Saranurak '19, Dory/Efron/Mukhopadhyay/Nanongkai '21]
  - Find greedy trees (easy).
  - Find cuts 2-respecting a tree (hard).

A closer look at the Min-Cut and Tree-Packings

- Consider a min-cut C.

A closer look at the Min-Cut and Tree-Packings

- Consider a min-cut C.
- The average number of times a tree in $\mathcal{T}$ crosses C is

$$\frac{1}{|\mathcal{T}|}\sum_{e\in C} L^{\mathcal{T}}(e) = \sum_{e\in C} \ell^{\mathcal{T}}(e).$$

A closer look at the Min-Cut and Tree-Packings

- Consider a min-cut C.
- The average number of times a tree in $\mathcal{T}$ crosses C is

$$\frac{1}{|\mathcal{T}|}\sum_{e\in C} L^{\mathcal{T}}(e) = \sum_{e\in C} \ell^{\mathcal{T}}(e).$$

- So if $\sum_{e\in C} \ell^{\mathcal{T}}(e) < 2$, at least one tree 1-respects C.

A closer look at the Min-Cut and Tree-Packings

- Consider a min-cut C.
- The average number of times a tree in $\mathcal{T}$ crosses C is

$$\frac{1}{|\mathcal{T}|}\sum_{e\in C} L^{\mathcal{T}}(e) = \sum_{e\in C} \ell^{\mathcal{T}}(e).$$

- So if $\sum_{e\in C} \ell^{\mathcal{T}}(e) < 2$, at least one tree 1-respects C.
- If $\sum_{e\in C} \ell^{*}(e) \ll 2$, then *crude* approximation gives $\sum_{e\in C} \ell^{\mathcal{T}}(e) < 2$.

A closer look at the Min-Cut and Tree-Packings

- Consider a min-cut C.
- The average number of times a tree in $\mathcal{T}$ crosses C is

$$\frac{1}{|\mathcal{T}|}\sum_{e\in C} L^{\mathcal{T}}(e) = \sum_{e\in C} \ell^{\mathcal{T}}(e).$$

- So if $\sum_{e\in C} \ell^{\mathcal{T}}(e) < 2$, at least one tree 1-respects C.
- If $\sum_{e\in C} \ell^{*}(e) \ll 2$, then *crude* approximation gives $\sum_{e\in C} \ell^{\mathcal{T}}(e) < 2$.
- Crude implies small $|\mathcal{T}|$ suffices.

A closer look at the Min-Cut and Tree-Packings

- Consider a min-cut C.
- The average number of times a tree in $\mathcal{T}$ crosses C is

$$\frac{1}{|\mathcal{T}|}\sum_{e\in C} L^{\mathcal{T}}(e) = \sum_{e\in C} \ell^{\mathcal{T}}(e).$$

- So if $\sum_{e\in C} \ell^{\mathcal{T}}(e) < 2$, at least one tree 1-respects C.
- If $\sum_{e\in C} \ell^{*}(e) \ll 2$, then *crude* approximation gives $\sum_{e\in C} \ell^{\mathcal{T}}(e) < 2$.
- Crude implies small $|\mathcal{T}|$ suffices.
- Left with the case that every min-cut C has $\sum_{e\in C} \ell^{*}(e) \approx 2$.

A closer look at the Min-Cut and Tree-Packings

- Consider a min-cut C.
- The average number of times a tree in $\mathcal{T}$ crosses C is

$$\frac{1}{|\mathcal{T}|}\sum_{e\in C} L^{\mathcal{T}}(e) = \sum_{e\in C} \ell^{\mathcal{T}}(e).$$

- So if $\sum_{e\in C} \ell^{\mathcal{T}}(e) < 2$, at least one tree 1-respects C.
- If $\sum_{e\in C} \ell^{*}(e) \ll 2$, then *crude* approximation gives $\sum_{e\in C} \ell^{\mathcal{T}}(e) < 2$.
- Crude implies small $|\mathcal{T}|$ suffices.
- Left with the case that every min-cut C has $\sum_{e\in C} \ell^{*}(e) \approx 2$.
- Every edge e participating in a min-cut has $\ell^{*}(e) \approx \frac{2}{\lambda}$.

A closer look at the Min-Cut and Tree-Packings

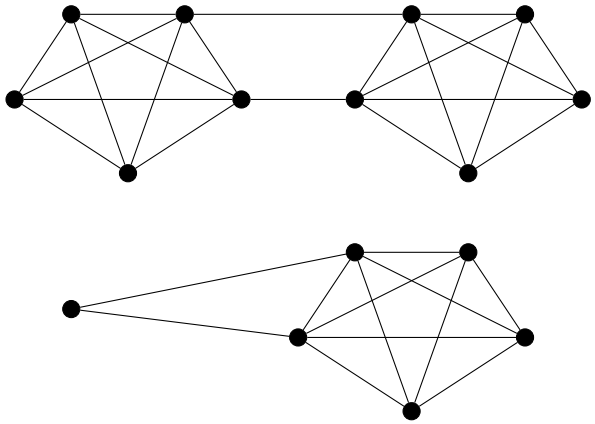
- Consider a min-cut C.
- The average number of times a tree in $\mathcal{T}$ crosses C is

$$\frac{1}{|\mathcal{T}|}\sum_{e\in C} L^{\mathcal{T}}(e) = \sum_{e\in C} \ell^{\mathcal{T}}(e).$$

- So if $\sum_{e\in C} \ell^{\mathcal{T}}(e) < 2$, at least one tree 1-respects C.
- If $\sum_{e\in C} \ell^{*}(e) \ll 2$, then *crude* approximation gives $\sum_{e\in C} \ell^{\mathcal{T}}(e) < 2$.
- Crude implies small $|\mathcal{T}|$ suffices.
- Left with the case that every min-cut C has $\sum_{e\in C} \ell^{*}(e) \approx 2$.
- Every edge e participating in a min-cut has $\ell^{*}(e) \approx \frac{2}{\lambda}$.
- All other edges f have $\ell^{*}(f) < \frac{2}{\lambda}$.

Contracted Graphs

- Every edge e participating in a min-cut has $\ell^*(e) \approx \frac{2}{\lambda}$.
- All other edges f have $\ell^*(f) < \frac{2}{\lambda}$.



Trivial Min-Cuts

- Left with the case that every min-cut C has $\sum_{e \in C} \ell^*(e) \approx 2$.
- Every edge e participating in a min-cut has $\ell^*(e) \approx \frac{2}{\lambda}$.

Trivial Min-Cuts

- Left with the case that every min-cut C has $\sum_{e \in C} \ell^*(e) \approx 2$.
- Every edge e participating in a min-cut has $\ell^*(e) \approx \frac{2}{\lambda}$.
- Claim 1: For some known $c \approx \frac{2}{\lambda}$, $G/\{e \in E : \ell^*(e) < c\}$ must contain trivial min-cuts.

Proof of Claim 1

Claim 1: For some $c \approx \frac{2}{\lambda}$, $G/\{e \in E : \ell^*(e) < c\}$ must contain trivial min-cuts.

Proof.

- Suppose not: min-deg $\geq \lambda + 1$.

□

Proof of Claim 1

Claim 1: For some $c \approx \frac{2}{\lambda}$, $G/\{e \in E : \ell^*(e) < c\}$ must contain trivial min-cuts.

Proof.

- Suppose not: min-deg $\geq \lambda + 1$.
- $G/\{e \in E : \ell^*(e) < c\}$ contains at least $\frac{1}{2}(\lambda + 1)|V(G/\{e \in E : \ell^*(e) < c\})|$ edges.

Proof of Claim 1

Claim 1: For some $c \approx \frac{2}{\lambda}$, $G/\{e \in E : \ell^*(e) < c\}$ must contain trivial min-cuts.

Proof.

- Suppose not: min-deg $\geq \lambda + 1$.
- $G/\{e \in E : \ell^*(e) < c\}$ contains at least $\frac{1}{2}(\lambda+1)|V(G/\{e \in E : \ell^*(e) < c\})|$ edges.
- $G/\{e \in E : \ell^*(e) < c\}$ are the edges with $\ell^*(e) \geq c \iff (\ell^*(e))^{-1} \leq c^{-1}$.

□

Proof of Claim 1

Claim 1: For some $c \approx \frac{2}{\lambda}$, $G/\{e \in E : \ell^*(e) < c\}$ must contain trivial min-cuts.

Proof.

- Suppose not: min-deg $\geq \lambda + 1$.
- $G/\{e \in E : \ell^*(e) < c\}$ contains at least $\frac{1}{2}(\lambda+1)|V(G/\{e \in E : \ell^*(e) < c\})|$ edges.
- $G/\{e \in E : \ell^*(e) < c\}$ are the edges with $\ell^*(e) \geq c \iff (\ell^*(e))^{-1} \leq c^{-1}$.
- For the minimizing partition $\mathcal{P}$ we have $\frac{|E(G/\mathcal{P})|}{|\mathcal{P}|-1} \leq (\ell^*(e))^{-1} \leq 1/c$.

□

Proof of Claim 1

Claim 1: For some $c \approx \frac{2}{\lambda}$, $G/\{e \in E : \ell^*(e) < c\}$ must contain trivial min-cuts.

Proof.

- Suppose not: min-deg $\geq \lambda + 1$.
- $G/\{e \in E : \ell^*(e) < c\}$ contains at least $\frac{1}{2}(\lambda+1)|V(G/\{e \in E : \ell^*(e) < c\})|$ edges.
- $G/\{e \in E : \ell^*(e) < c\}$ are the edges with $\ell^*(e) \geq c \iff (\ell^*(e))^{-1} \leq c^{-1}$.
- For the minimizing partition $\mathcal{P}$ we have $\frac{|E(G/\mathcal{P})|}{|\mathcal{P}|-1} \leq (\ell^*(e))^{-1} \leq 1/c$.
- On each level, $E(G/\{e \in E : \ell^*(e) < c)\}$ corresponds to the minimum partition.

□

Proof of Claim 1

Claim 1: For some $c \approx \frac{2}{\lambda}$, $G/\{e \in E : \ell^*(e) < c\}$ must contain trivial min-cuts.

Proof.

- Suppose not: min-deg $\geq \lambda + 1$.
- $G/\{e \in E : \ell^*(e) < c\}$ contains at least $\frac{1}{2}(\lambda+1)|V(G/\{e \in E : \ell^*(e) < c\})|$ edges.
- $G/\{e \in E : \ell^*(e) < c\}$ are the edges with $\ell^*(e) \geq c \iff (\ell^*(e))^{-1} \leq c^{-1}$.
- For the minimizing partition $\mathcal{P}$ we have $\frac{|E(G/\mathcal{P})|}{|\mathcal{P}|-1} \leq (\ell^*(e))^{-1} \leq 1/c$.
- On each level, $E(G/\{e \in E : \ell^*(e) < c)\}$ corresponds to the minimum partition.
- We get $|E(G/\{e \in E : \ell^*(e) < c)\} \leq c^{-1}(|V(G/\{e \in E : \ell^*(e) < c\})| - 1) \approx \frac{\lambda}{2}|V(G/\{e \in E : \ell^*(e) < c\})|$.

□

Proof of Claim 1

Claim 1: For some $c \approx \frac{2}{\lambda}$, $G/\{e \in E : \ell^*(e) < c\}$ must contain trivial min-cuts.

Proof.

- Suppose not: min-deg $\geq \lambda + 1$.
- $G/\{e \in E : \ell^*(e) < c\}$ contains at least $\frac{1}{2}(\lambda+1)|V(G/\{e \in E : \ell^*(e) < c\})|$ edges.
- $G/\{e \in E : \ell^*(e) < c\}$ are the edges with $\ell^*(e) \geq c \iff (\ell^*(e))^{-1} \leq c^{-1}$.
- For the minimizing partition $\mathcal{P}$ we have $\frac{|E(G/\mathcal{P})|}{|\mathcal{P}|-1} \leq (\ell^*(e))^{-1} \leq 1/c$.
- On each level, $E(G/\{e \in E : \ell^*(e) < c)\}$ corresponds to the minimum partition.
- We get $|E(G/\{e \in E : \ell^*(e) < c)\} \leq c^{-1}(|V(G/\{e \in E : \ell^*(e) < c\})| - 1) \approx \frac{\lambda}{2}|V(G/\{e \in E : \ell^*(e) < c\})|$.
- Contradiction.

□

Trivial Min-Cuts

- Left with the case that every min-cut C has $\sum_{e \in C} \ell^*(e) \approx 2$.
- Every edge e participating in a min-cut has $\ell^*(e) \approx \frac{2}{\lambda}$.
- Claim 1: For some known $c \approx \frac{2}{\lambda}$, $G/\{e \in E : \ell^*(e) < c\}$ must contain trivial min-cuts.

Trivial Min-Cuts

- Left with the case that every min-cut C has $\sum_{e \in C} \ell^*(e) \approx 2$.
- Every edge e participating in a min-cut has $\ell^*(e) \approx \frac{2}{\lambda}$.
- Claim 1: For some known $c \approx \frac{2}{\lambda}$, $G/\{e \in E : \ell^*(e) < c\}$ must contain trivial min-cuts.
- Claim 2: For $|\mathcal{T}| = \Theta(\lambda_{\max}^3 \log m)$, all edges e in a min-cut have $\ell^{\mathcal{T}}(e) > c$ and all edges in $G[X]$ have $\ell^{\mathcal{T}}(e) < c$.

Trivial Min-Cuts

- Left with the case that every min-cut C has $\sum_{e \in C} \ell^*(e) \approx 2$.
- Every edge e participating in a min-cut has $\ell^*(e) \approx \frac{2}{\lambda}$.
- Claim 1: For some known $c \approx \frac{2}{\lambda}$, $G/\{e \in E : \ell^*(e) < c\}$ must contain trivial min-cuts.
- Claim 2: For $|\mathcal{T}| = \Theta(\lambda_{\max}^3 \log m)$, all edges e in a min-cut have $\ell^{\mathcal{T}}(e) > c$ and all edges in $G[X]$ have $\ell^{\mathcal{T}}(e) < c$.
- So a trivial min-cut of $G/\{e \in E : \ell^*(e) < c\}$ is a trivial min-cut of $G/\{e \in E : \ell^{\mathcal{T}}(e) < c\}$.

Algorithm

- Maintain tree-packing $\mathcal{T}$ of size $|\mathcal{T}| = \Theta(\lambda_{\max}^3 \log m)$.
  - Decreased recourse ($\lambda_{\max}^5$ instead of $\lambda_{\max}^6$).
- Maintain 1-respecting cuts for each $T \in \mathcal{T}$ in $\tilde{O}(\sqrt{\lambda_{\max} n})$ worst-case update time.
- Maintain trivial cuts in contracted graph $G/\{e \in E : \ell^{\mathcal{T}}(e) < c\}$ using top trees.

Algorithm

- Maintain tree-packing $\mathcal{T}$ of size $|\mathcal{T}| = \Theta(\lambda_{\max}^3 \log m)$.
  - Decreased recourse ($\lambda_{\max}^5$ instead of $\lambda_{\max}^6$).
- Maintain 1-respecting cuts for each $T \in \mathcal{T}$ in $\tilde{O}(\sqrt{\lambda_{\max} n})$ worst-case update time.
- Maintain trivial cuts in contracted graph $G/\{e \in E : \ell^{\mathcal{T}}(e) < c\}$ using top trees.

Theorem

Unweighted, undirected (multi-)graph $G = (V, E)$. There exists a deterministic dynamic algorithm that maintains the exact min-cut value λ if $\lambda \leq \lambda_{\max}$ in $\tilde{O}(\lambda_{\max}^{5.5}\sqrt{n})$ worst-case update time.

Previously: $\tilde{O}(\lambda_{\max}^{14.5}\sqrt{n})$ [Thorup '07].

Part 1: Dynamic Min-Cut

Part 2: Dynamic Arboricity

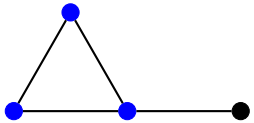
Part 3: More on Tree-Packings

Arboricity

Define

- The fractional arboricity:

$$\alpha := \max_{S \subseteq V} \frac{|E(S)|}{|S| - 1}.$$



Arboricity

Define

- The fractional arboricity:

$$\alpha := \max_{S \subseteq V} \frac{|E(S)|}{|S| - 1}.$$

- The (integral) arboricity: $\tilde{\alpha}$ the minimum number of trees that cover E.

Arboricity

Define

- The fractional arboricity:

$$\alpha := \max_{S \subseteq V} \frac{|E(S)|}{|S| - 1}.$$

- The (integral) arboricity: $\tilde{\alpha}$ the minimum number of trees that cover E.

Theorem. We have $\tilde{\alpha} = \lceil \alpha \rceil$.

Arboricity

Theorem

Unweighted, undirected (multi-)graph $G = (V, E)$. There exists a deterministic dynamic algorithm that maintains a $(1+\varepsilon)$-approximation of the fractional arboricity α when $\alpha \leq \alpha_{\max}$ in $O(\alpha_{\max} \log^6 m/\varepsilon^4)$ amortized update time.

Arboricity

Theorem

Unweighted, undirected (multi-)graph $G = (V, E)$. There exists a deterministic dynamic algorithm that maintains a $(1+\varepsilon)$-approximation of the fractional arboricity α when $\alpha \leq \alpha_{\max}$ in $O(\alpha_{\max} \log^6 m/\varepsilon^4)$ amortized update time.

- Previously only $(2+\varepsilon)$-approximation.
- Worst-case (Las Vegas) with $m^{o(1)}$.
- We showed that $\alpha = \frac{1}{\min_{e \in E} \ell^*(e)}$ and use a greedy tree-packing.
  - [Cen/Fleischmann/Li/Li/Panigrahi '25] Used this fact to compute static arboricity faster
- Additional tricks to lower the recourse.

Arboricity

Fully dynamic $(1+\varepsilon)$-approximate arboricity

- Deterministic $\sim \alpha \leq \sqrt{m}$ update time.

Arboricity

Fully dynamic $(1+\varepsilon)$-approximate arboricity

- Deterministic $\sim \alpha \leq \sqrt{m}$ update time.
- Simple graphs (deterministic): $\text{poly}(\log n, \varepsilon^{-1})$ amortized
  - Uses out-orientations for large values [Chekuri/Christiansen/Holm/van der Hoog/Quanrud/Rotenberg/Schwiegelshohn '24].

Arboricity

Fully dynamic $(1+\varepsilon)$-approximate arboricity

- Deterministic $\sim \alpha \leq \sqrt{m}$ update time.
- Simple graphs (deterministic): $\text{poly}(\log n, \varepsilon^{-1})$ amortized
  - Uses out-orientations for large values [Chekuri/Christiansen/Holm/van der Hoog/Quanrud/Rotenberg/Schwiegelshohn '24].

Multi-Graphs

- Oblivious adversary: $\text{poly}(\log n, \varepsilon^{-1})$ amortized update time
  - Standard uniform sampling $\sim \frac{\log n}{\alpha \varepsilon^2}$.

Arboricity

Fully dynamic $(1+\varepsilon)$-approximate arboricity

- Deterministic $\sim \alpha \leq \sqrt{m}$ update time.
- Simple graphs (deterministic): $\text{poly}(\log n, \varepsilon^{-1})$ amortized
  - Uses out-orientations for large values [Chekuri/Christiansen/Holm/van der Hoog/Quanrud/Rotenberg/Schwiegelshohn '24].

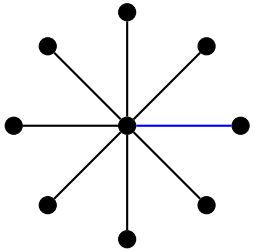
Multi-Graphs

- Oblivious adversary: $\text{poly}(\log n, \varepsilon^{-1})$ amortized update time
  - Standard uniform sampling $\sim \frac{\log n}{\alpha \varepsilon^2}$.
- Adaptive adversary: $\text{poly}(\log n, \varepsilon^{-1})$ amortized update time
  - Fancy sampling.

Fancy Sampling

Uniform sampling $\sim \frac{\log n}{\alpha \varepsilon^2}$.

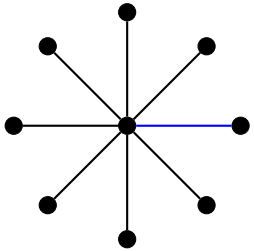
- Problem: Adaptive adversary can delete sampled edges.

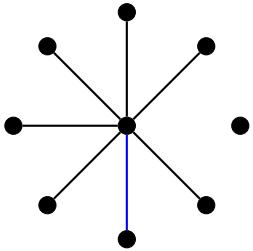


Fancy Sampling

Uniform sampling $\sim \frac{\log n}{\alpha \varepsilon^2}$.

- Problem: Adaptive adversary can delete sampled edges.
- Solution: A vertex resamples its out-edges when affected.

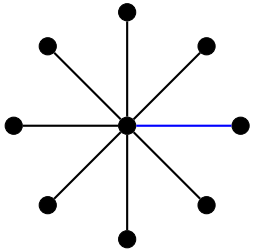


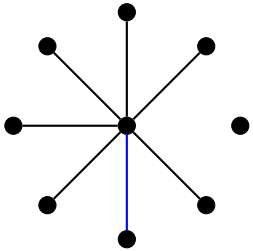


Fancy Sampling

Uniform sampling $\sim \frac{\log n}{\alpha \varepsilon^2}$.

- Problem: Adaptive adversary can delete sampled edges.
- Solution: A vertex resamples its out-edges when affected.
- Problem: Maximum degree $n - 1$.
  - In sampled graph $\approx n/\alpha$.

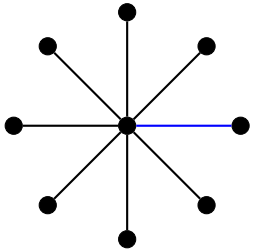


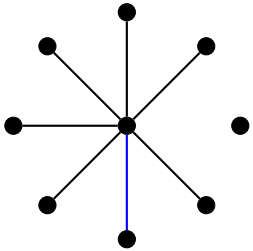


Fancy Sampling

Uniform sampling $\sim \frac{\log n}{\alpha\varepsilon^2}$.

- Problem: Adaptive adversary can delete sampled edges.
- Solution: A vertex resamples its out-edges when affected.
- Problem: Maximum degree $n-1$.
  - In sampled graph $\approx n/\alpha$.
- Solution: Use out-orientations such that each vertex 'owns' $O(\alpha)$ edges.
  - In sampled graph $O(\alpha \cdot \frac{\log n}{\alpha\varepsilon^2}) = O(\frac{\log n}{\varepsilon^2})$.





Part 1: Dynamic Min-Cut
Part 2: Dynamic Arboricity

Part 3: More on Tree-Packings

More on Tree-Packings

- Unweighted, undirected (multi-)graph $G = (V, E)$.
- Goal: $|\ell^{\mathcal{T}}(e) - \ell^*(e)| \leq \varepsilon/\lambda$ for all $e \in E$.
- [Thorup '07] $|\mathcal{T}| = O(\log m \cdot \lambda/\varepsilon^2)$ greedy trees.
- Can we do better?

More on Tree-Packings

- Unweighted, undirected (multi-)graph $G = (V, E)$.
- Goal: $|\ell^{\mathcal{T}}(e) - \ell^*(e)| \leq \varepsilon/\lambda$ for all $e \in E$.
- [Thorup '07] $|\mathcal{T}| = O(\log m \cdot \lambda/\varepsilon^2)$ greedy trees.
- Can we do better?

Theorem

In general, such a greedy tree-packing $\mathcal{T}$ needs $|\mathcal{T}| = \Omega(\lambda/\varepsilon^{3/2})$ trees, whenever $\varepsilon^{-1} = O(n^{1/3})$.

More on Tree-Packings

- Unweighted, undirected (multi-)graph $G = (V, E)$.
- Goal: $|\ell^{\mathcal{T}}(e) - \ell^*(e)| \leq \varepsilon/\lambda$ for all $e \in E$.
- [Thorup '07] $|\mathcal{T}| = O(\log m \cdot \lambda/\varepsilon^2)$ greedy trees.
- Can we do better?

Theorem

In general, such a greedy tree-packing $\mathcal{T}$ needs $|\mathcal{T}| = \Omega(\lambda/\varepsilon^{3/2})$ trees, whenever $\varepsilon^{-1} = O(n^{1/3})$.

[Arkhipov/Kolmogorov '25]: Need $|\mathcal{T}| = \Omega(\lambda/\varepsilon^2)$ trees

More on Tree-Packings

- Unweighted, undirected (multi-)graph $G = (V, E)$.
- Goal: $|\ell^{\mathcal{T}}(e) - \ell^*(e)| \leq \varepsilon/\lambda$ for all $e \in E$.
- [Thorup '07] $|\mathcal{T}| = O(\log m \cdot \lambda/\varepsilon^2)$ greedy trees.
- Can we do better?

Theorem

In general, such a greedy tree-packing $\mathcal{T}$ needs $|\mathcal{T}| = \Omega(\lambda/\varepsilon^{3/2})$ trees, whenever $\varepsilon^{-1} = O(n^{1/3})$.

[Arkhipov/Kolmogorov '25]: Need $|\mathcal{T}| = \Omega(\lambda/\varepsilon^2)$ trees

Theorem

There exists such a tree-packing $\mathcal{T}$ with $|\mathcal{T}| = \Theta(\lambda/\varepsilon)$ trees.

More on Tree-Packings

- Unweighted, undirected (multi-)graph $G = (V, E)$.
- Goal: $|\ell^{\mathcal{T}}(e) - \ell^*(e)| \leq \varepsilon/\lambda$ for all $e \in E$.
- [Thorup '07] $|\mathcal{T}| = O(\log m \cdot \lambda/\varepsilon^2)$ greedy trees.
- Can we do better?

Theorem (joint work with Mara Grilnberger)

Let $\gamma \in [\lambda, \alpha]$ and let $\mathcal{T}$ be a greedy tree-packing with $|\mathcal{T}| \geq 3\gamma \cdot \log n/\varepsilon^2$. Then

$$|\ell^{\mathcal{T}}(e) - \ell^*(e)| \leq \varepsilon \ell^*(e)$$

for all $e \in E$ with $\ell^*(e) \geq 1/\gamma$ and

$$|\ell^{\mathcal{T}}(e) - \ell^*(e)| \leq \varepsilon/\gamma$$

for all $e \in E$ with $\ell^*(e) \leq 1/\gamma$.

In particular: need $\Omega(\alpha \cdot \log n/\varepsilon^2)$ trees to approximate the arboricity instead of $\Omega(\alpha^2 \cdot \log n/\varepsilon^2)$ trees.

Conclusion

Contributions

1. New insight on relation between tree-packings and min-cut.
2. First algorithms for dynamic $(1 \pm \varepsilon)$-approximate arboricity via tree-packings.
3. Subroutines for decreased recourse and better sampling.

Conclusion

Contributions

1. New insight on relation between tree-packings and min-cut.
2. First algorithms for dynamic $(1 \pm \varepsilon)$-approximate arboricity via tree-packings.
3. Subroutines for decreased recourse and better sampling.

Questions

- Can we use different tree-packings?
- Can we leverage these three contributions in other models?

Conclusion

Contributions

1. New insight on relation between tree-packings and min-cut.
2. First algorithms for dynamic $(1 \pm \varepsilon)$-approximate arboricity via tree-packings.
3. Subroutines for decreased recourse and better sampling.

Questions

- Can we use different tree-packings?
- Can we leverage these three contributions in other models?

Thank you!