Frontiers of Graph Algorithms

Fall 2023

Brown University

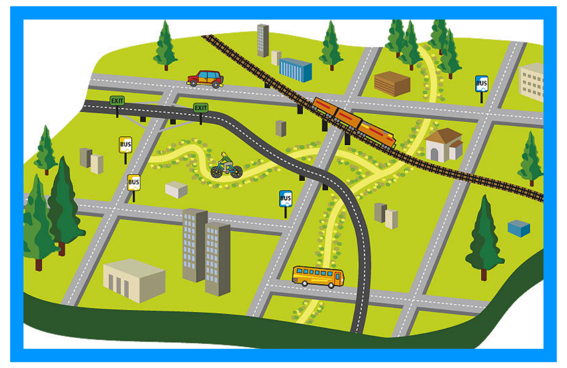
https://dhershko.github.io/teaching/fall23Seminar.html

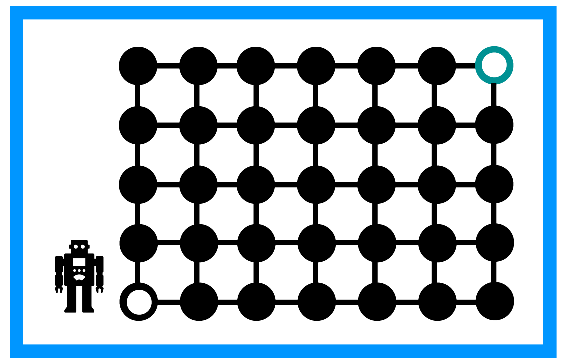
D Ellis Hershkowitz (Ellis)

Graph Algorithms

Why Study Graph (Algorithms)?

- General

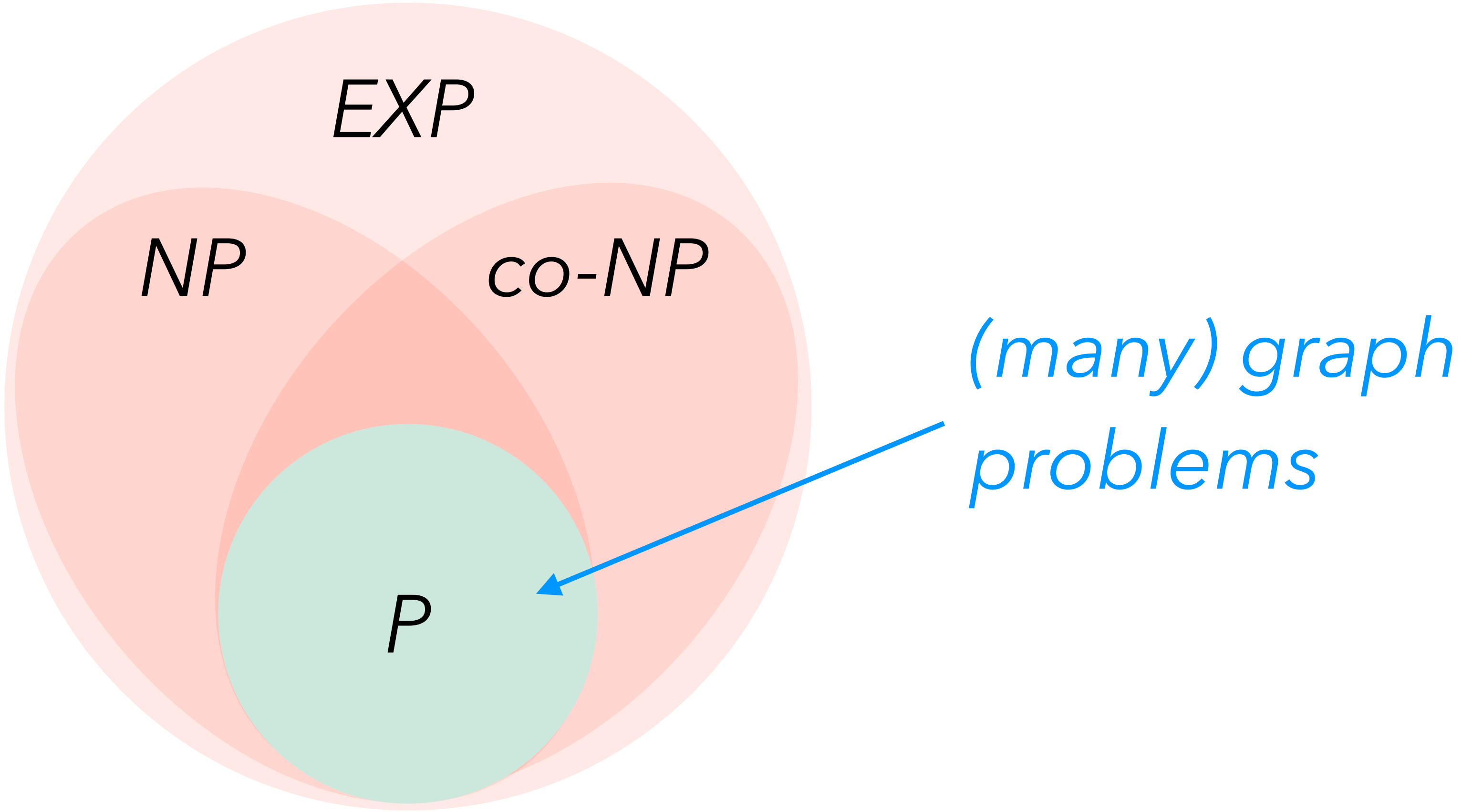


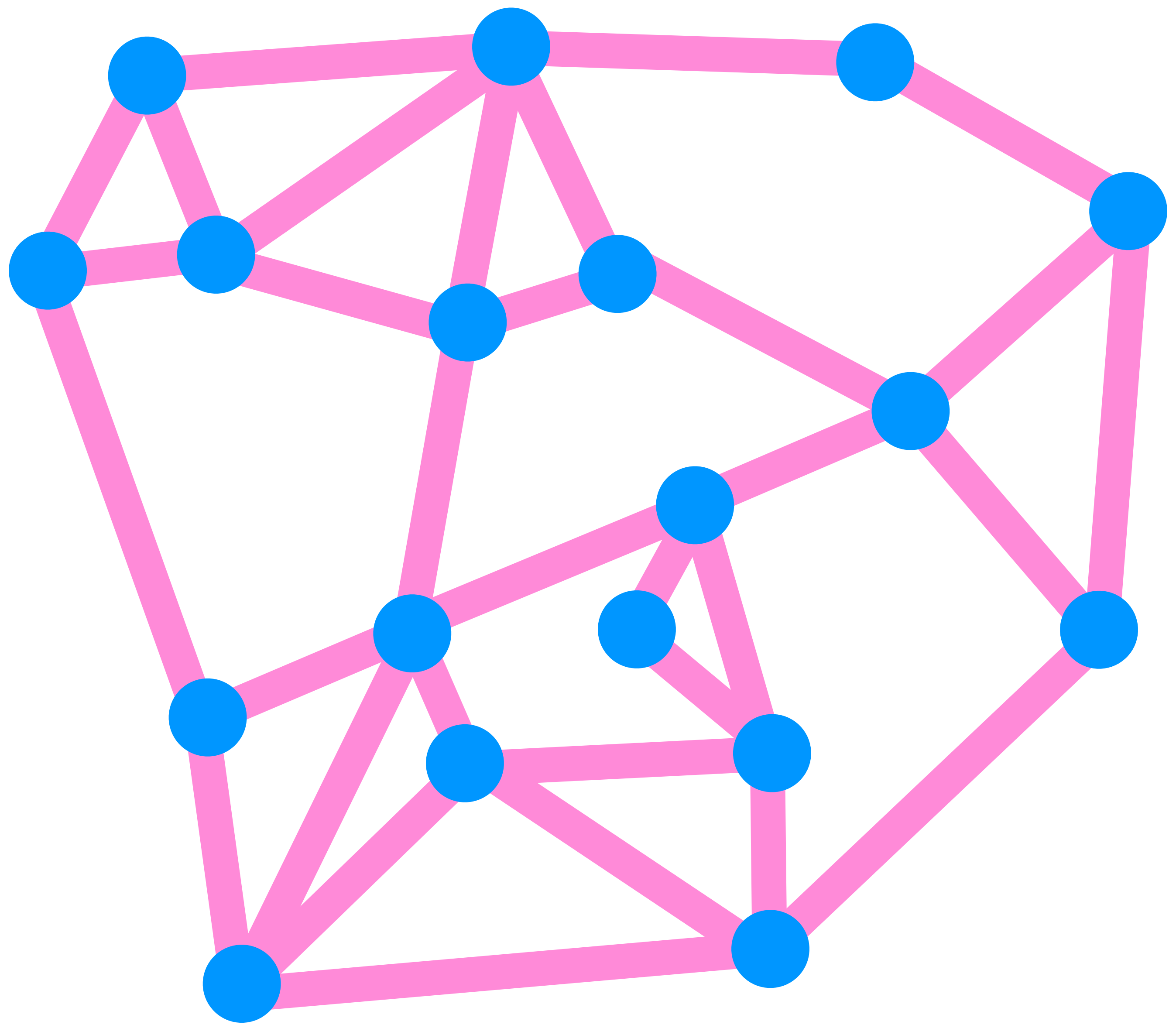


$\sum = \Pi$

…

- Computationally tractable





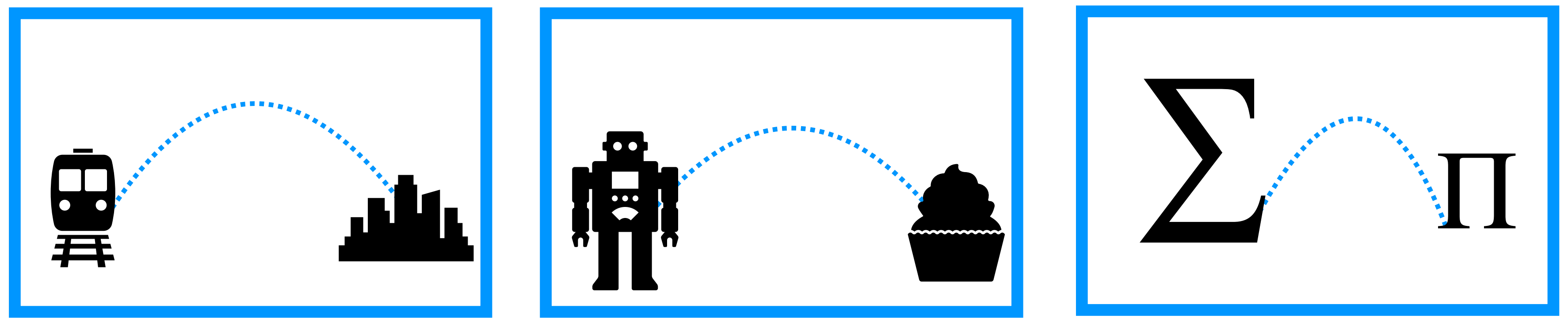
graph $G = (V, E)$

$n := |V|$ and $m := |E|$

Graph Algorithms

Why Study Graph Distance (Algorithms)?

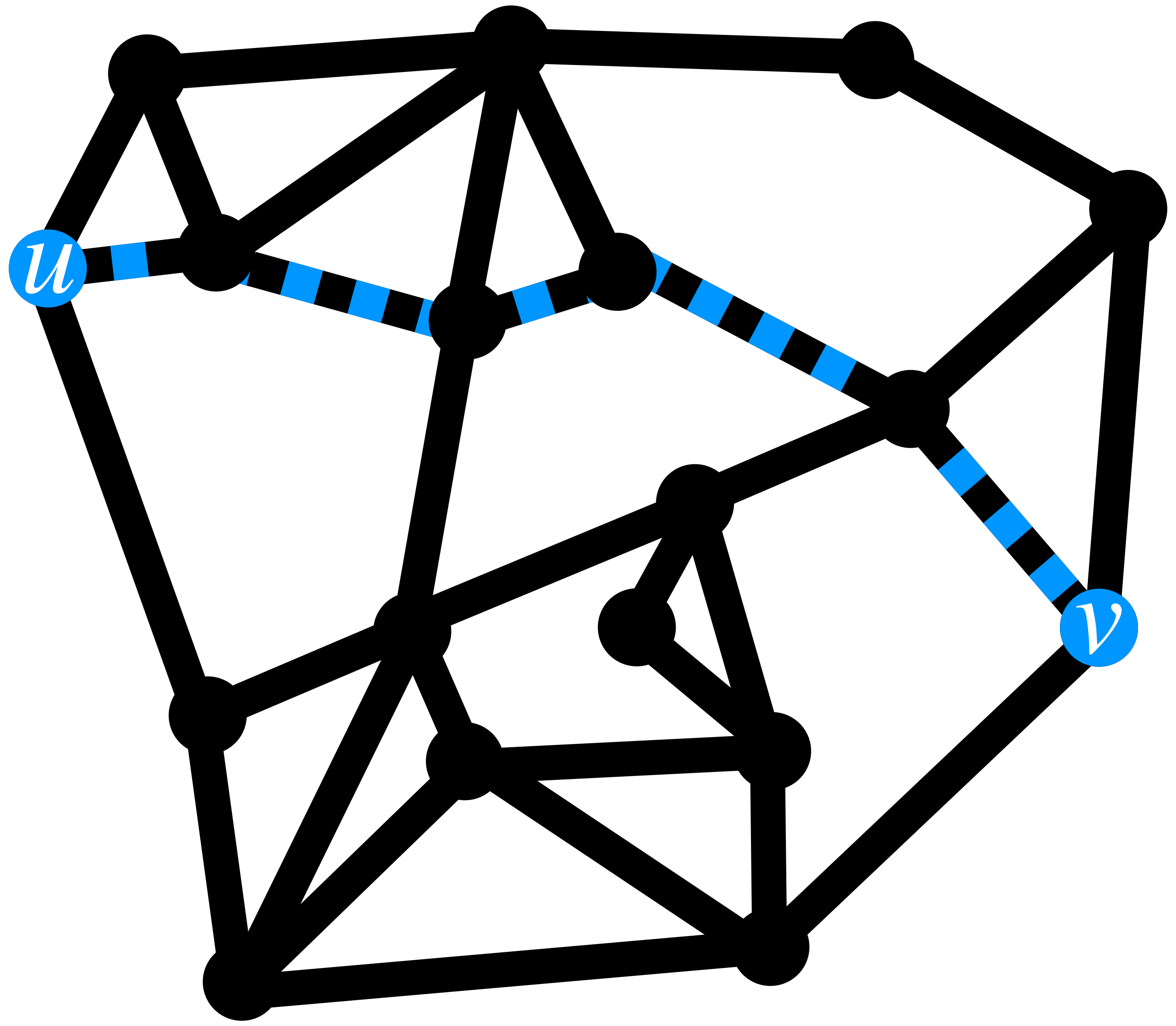
- General



…

- Computationally tractable

$d_G(u, v)$ for all u, v
in $O(n \cdot m)$ time



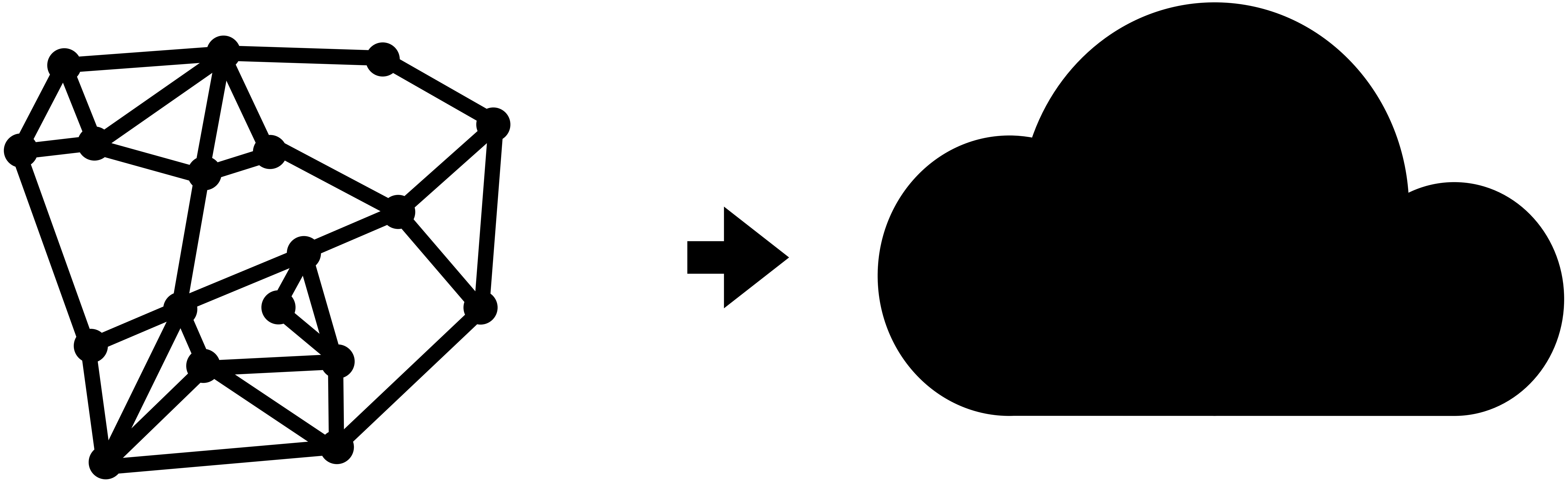
graph $G = (V, E)$

$(d_G(u, v) = 5)$

$d_G(u, v) := \min\{ |P| : \text{path } P \text{ from } u \text{ to } v\}$

Class Topic

Graph Sparsification



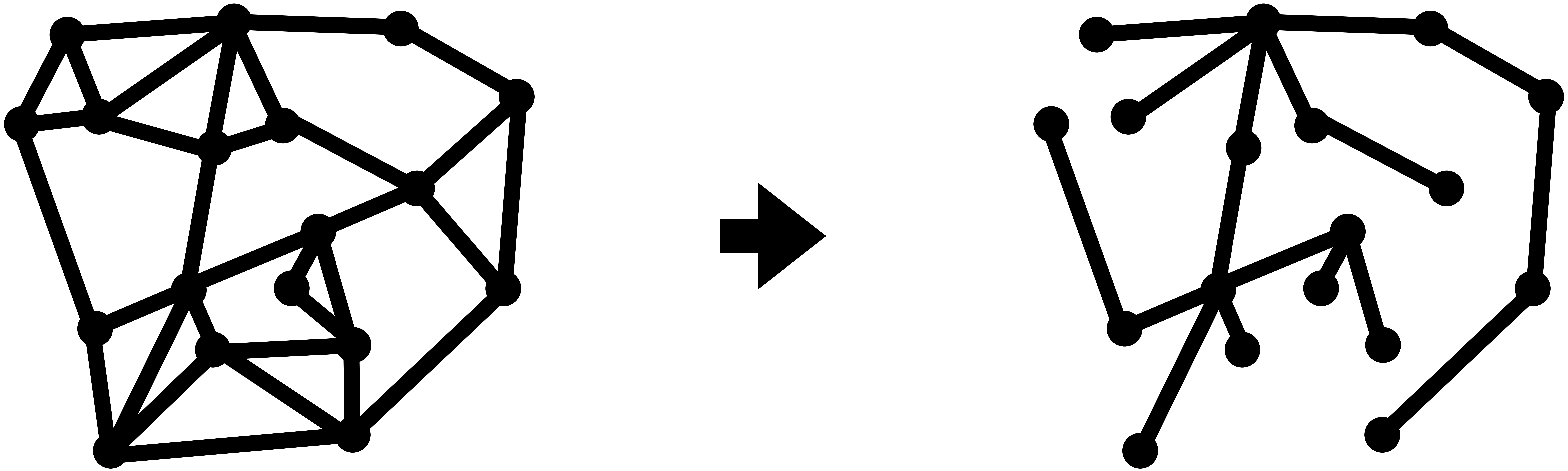
graph $G = (V, E)$

simple representation H
of some property of G

Focus: graph sparsification

Class Topic

Graph Distance Sparsification



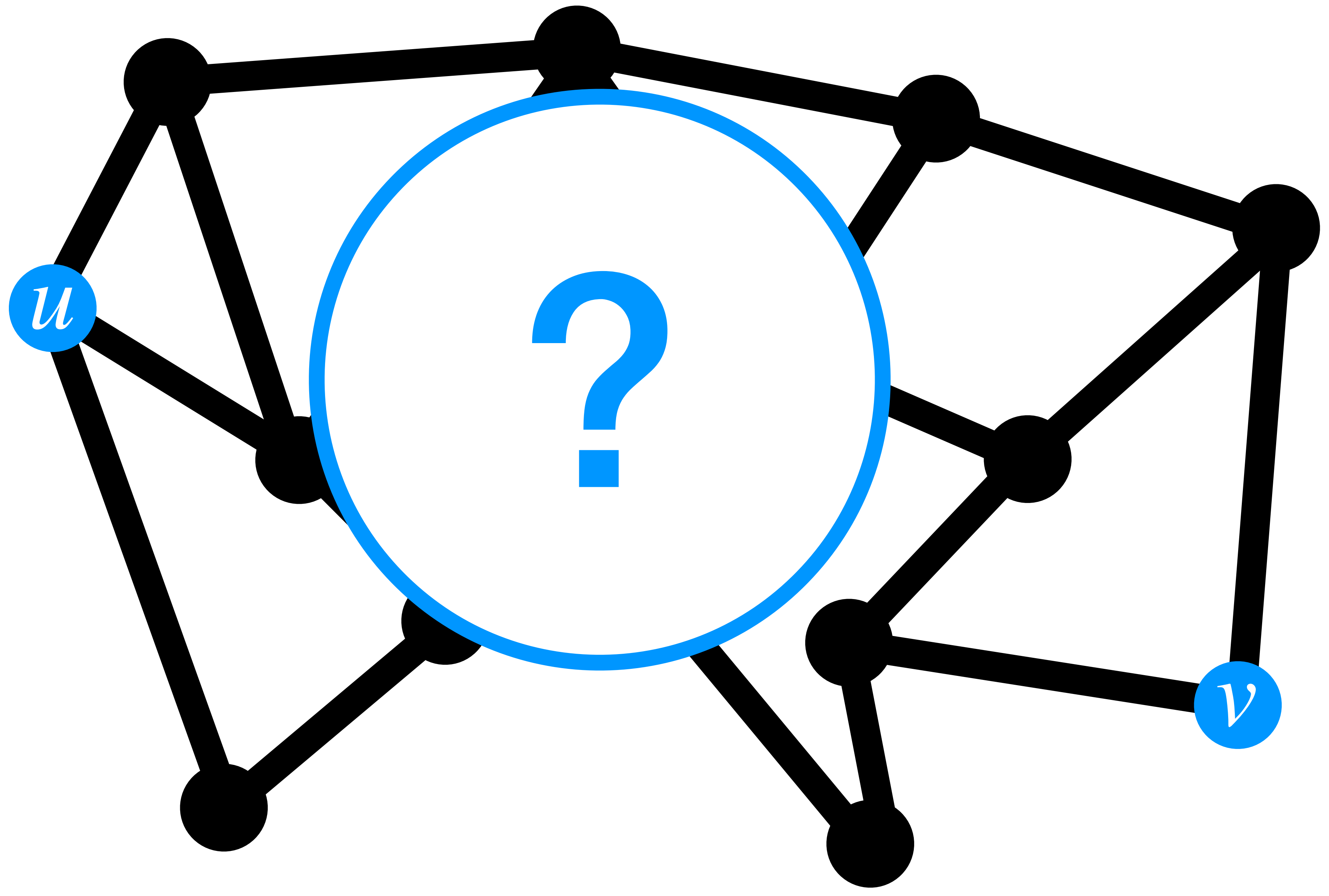
graph $G = (V, E)$

spanning tree H
s.t. $d_G = d_H$

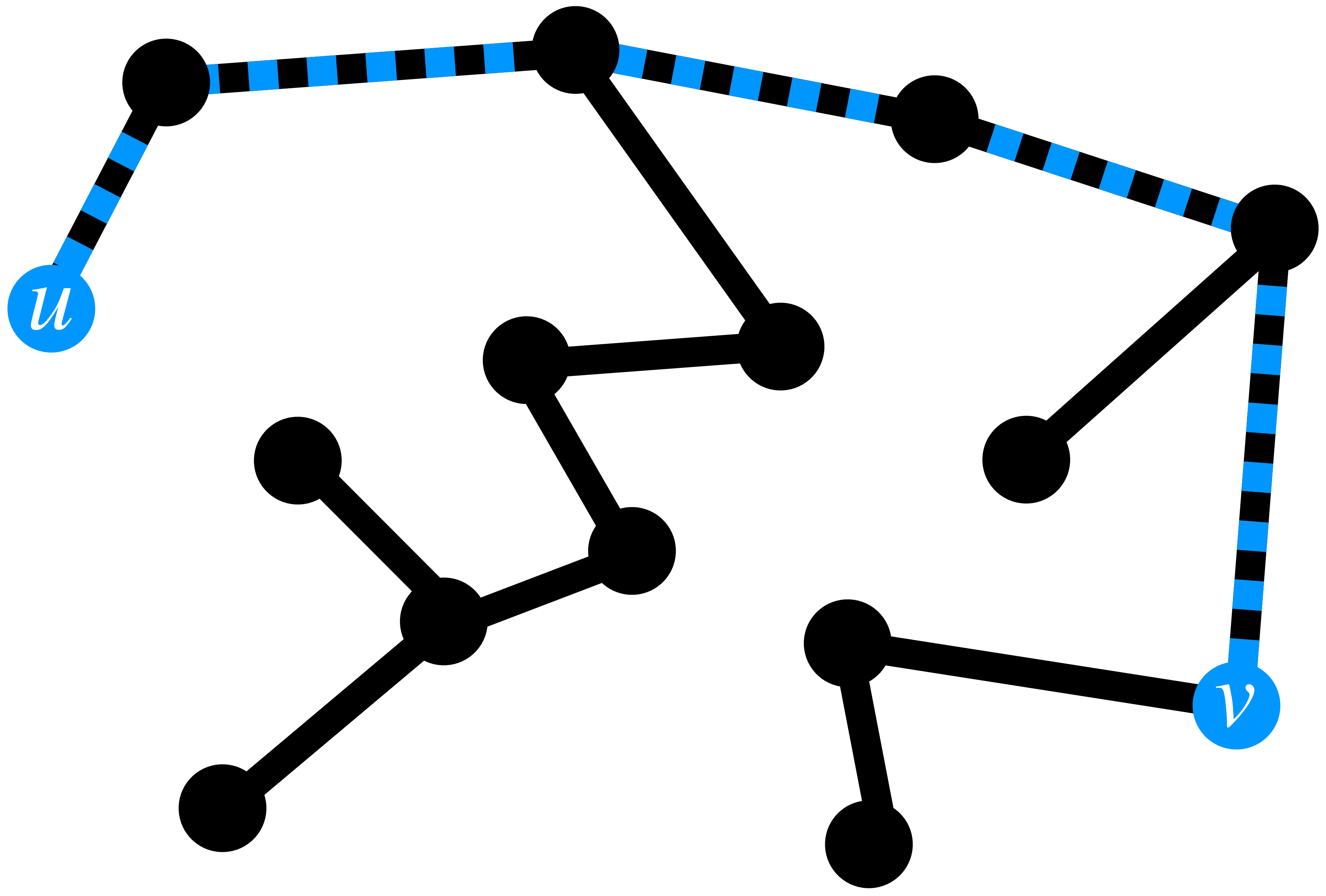
Focus: graph sparsification

Why Study Sparsification?

Theme 1: Sparsification Helps Algorithms



What's the $u \to v$ shortest path?

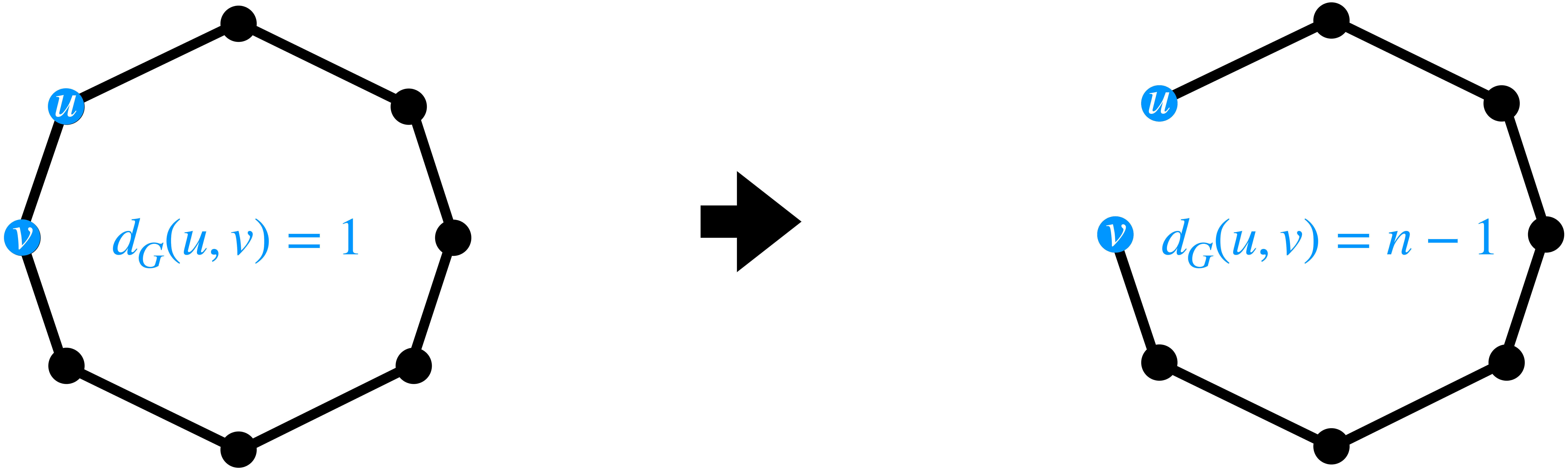


What's the $u \to v$ shortest path?

Focus: graph sparsification

Challenges of Sparsification

Theme 2: Approximation Helps Sparsification



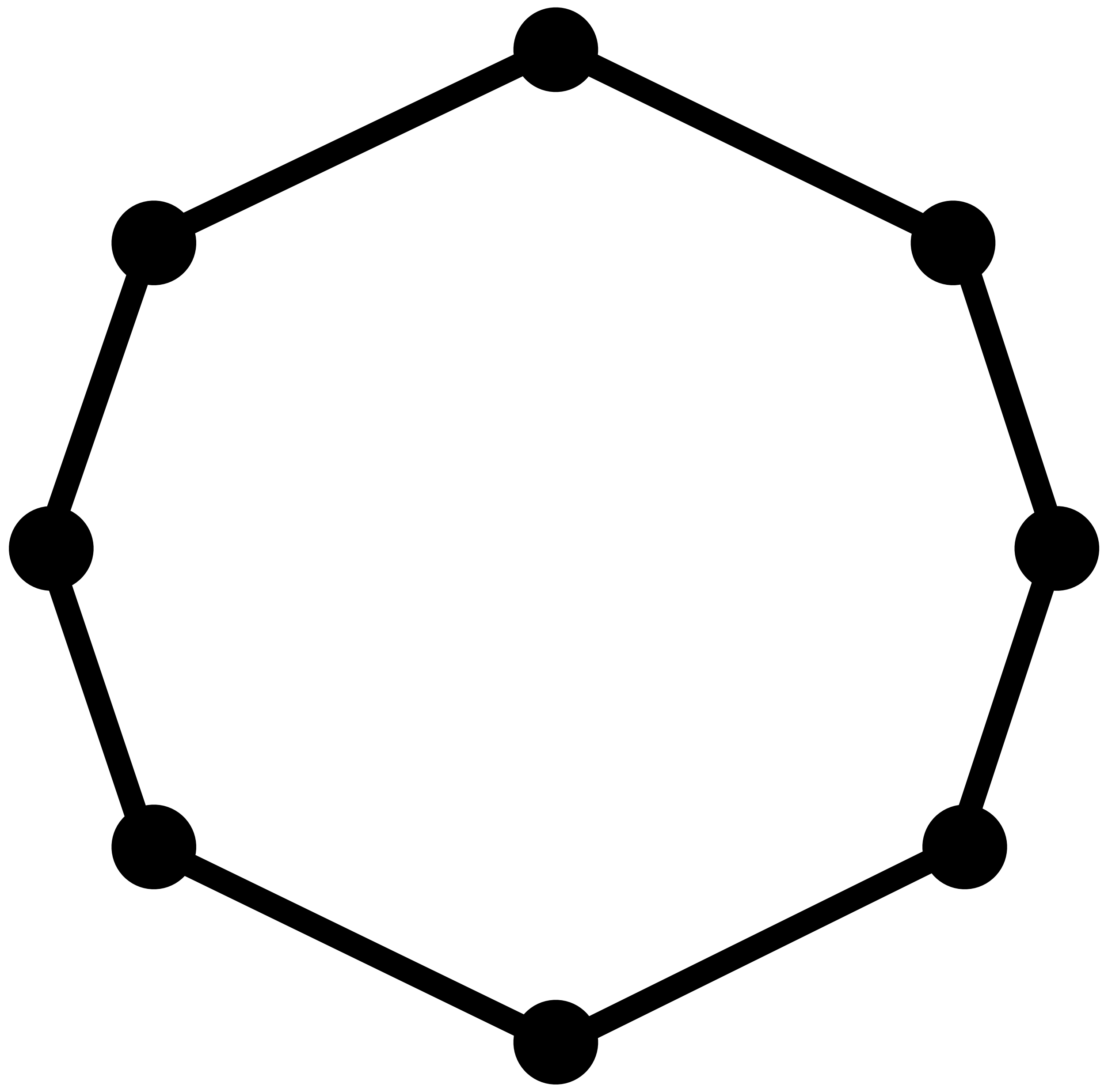
graph $G = (V, E)$

spanning tree H
s.t. $d_H = d_G$

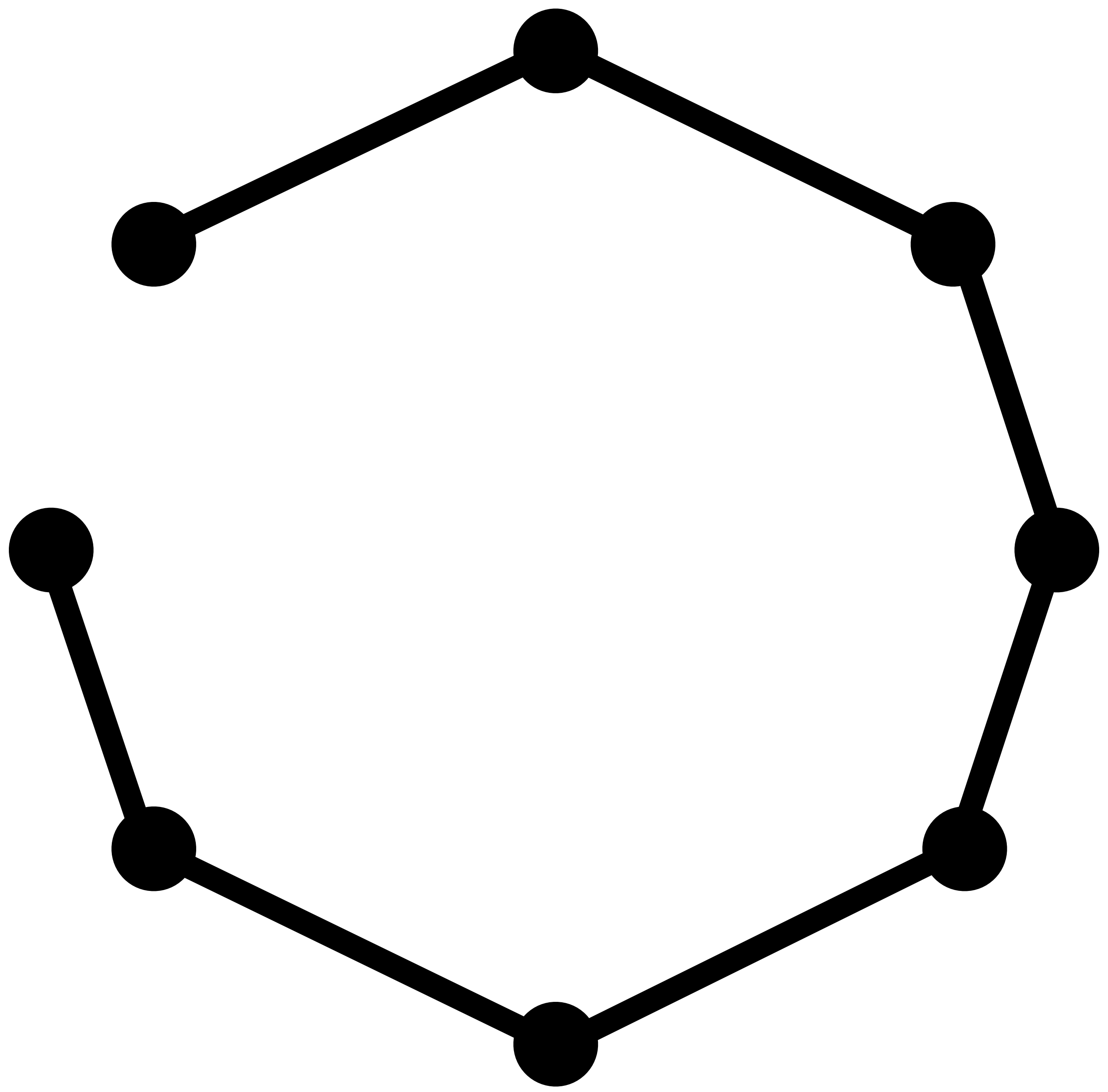
Focus: graph sparsification

Challenges of Sparsification

Theme 2: Approximation Helps Sparsification



graph $G = (V, E)$

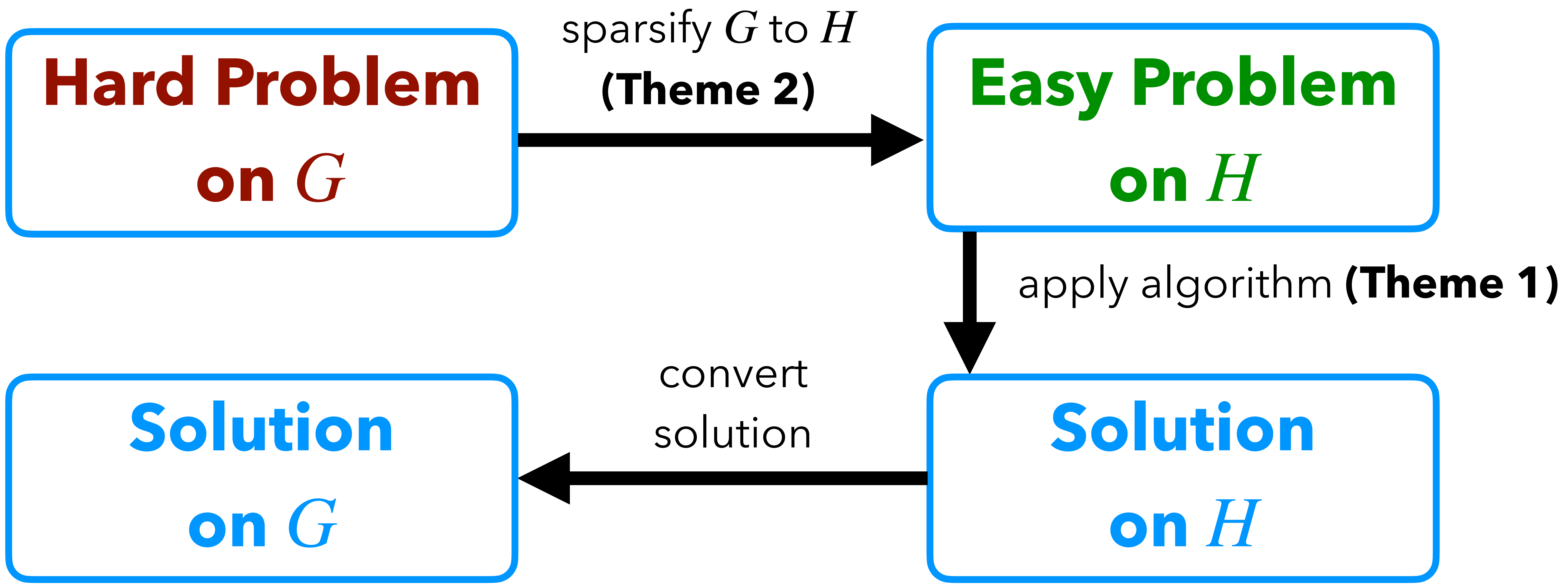


spanning *tree* H

s.t. $d_H \approx d_G$

Focus: graph sparsification

How To Solve Your Favorite Graph Problem



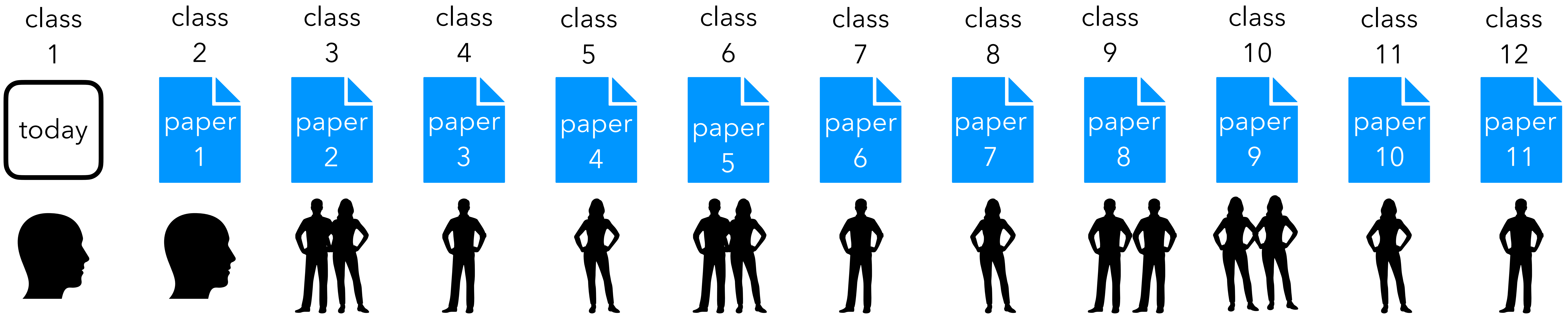
Focus: graph sparsification

Logistics Overview

Format Of Class

Seminar Format

- 11 (remaining) classes
- 1 paper / class (papers already chosen by me)
- First 2 classes by me
- For other classes 1-2 students present / class

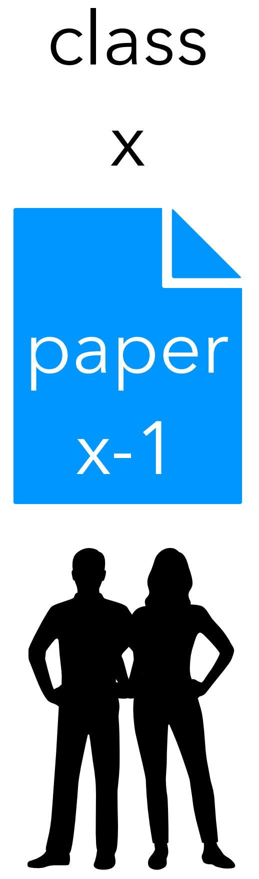


Format Of Class

Class Format

1. **Introduction:** ~30 minutes
2. **Break:** ~20 minutes
3. **Technical Details:** ~60 minutes
4. **Class Feedback:** ~15 minutes

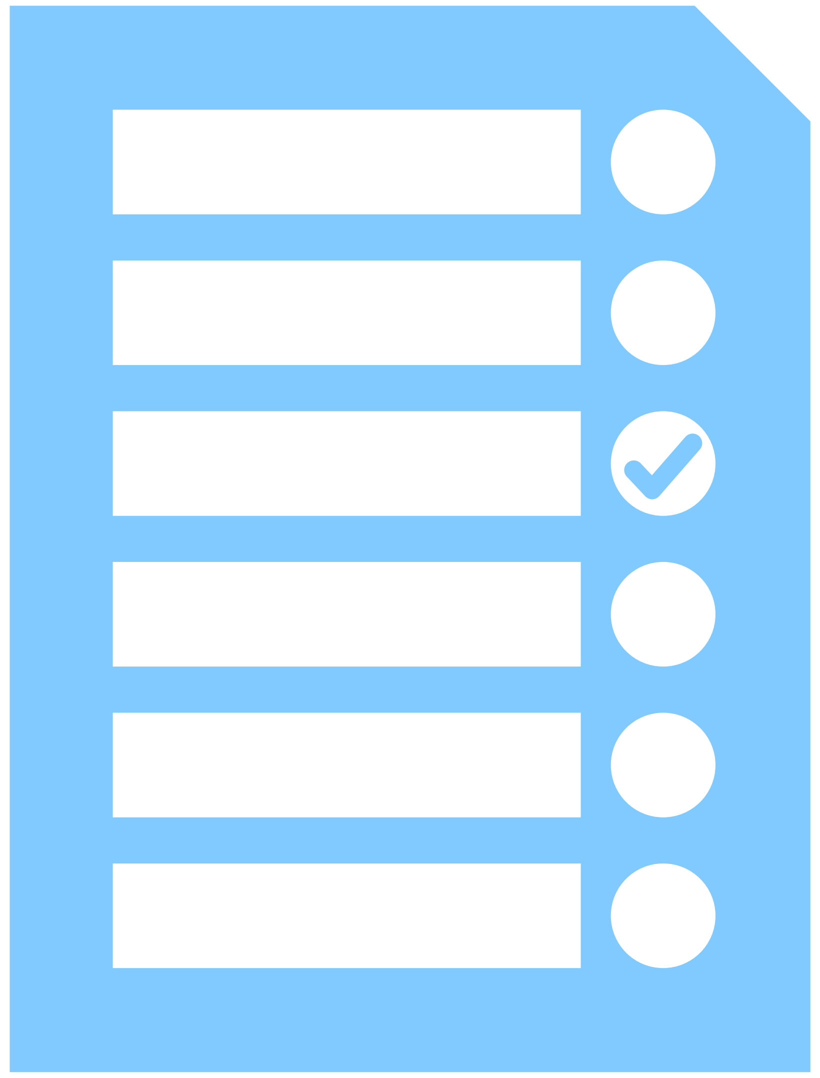
(flexible)



Format Of Class

Your Responsibilities

1. Fill out form of top 3 papers after shopping (**need Sep 20, 27 speakers now**)
2. Read your assigned paper
3. Prepare talk on paper + 6 questions
4. Practice (first half of) talk with me week before
5. Actively participate / give feedback after talks



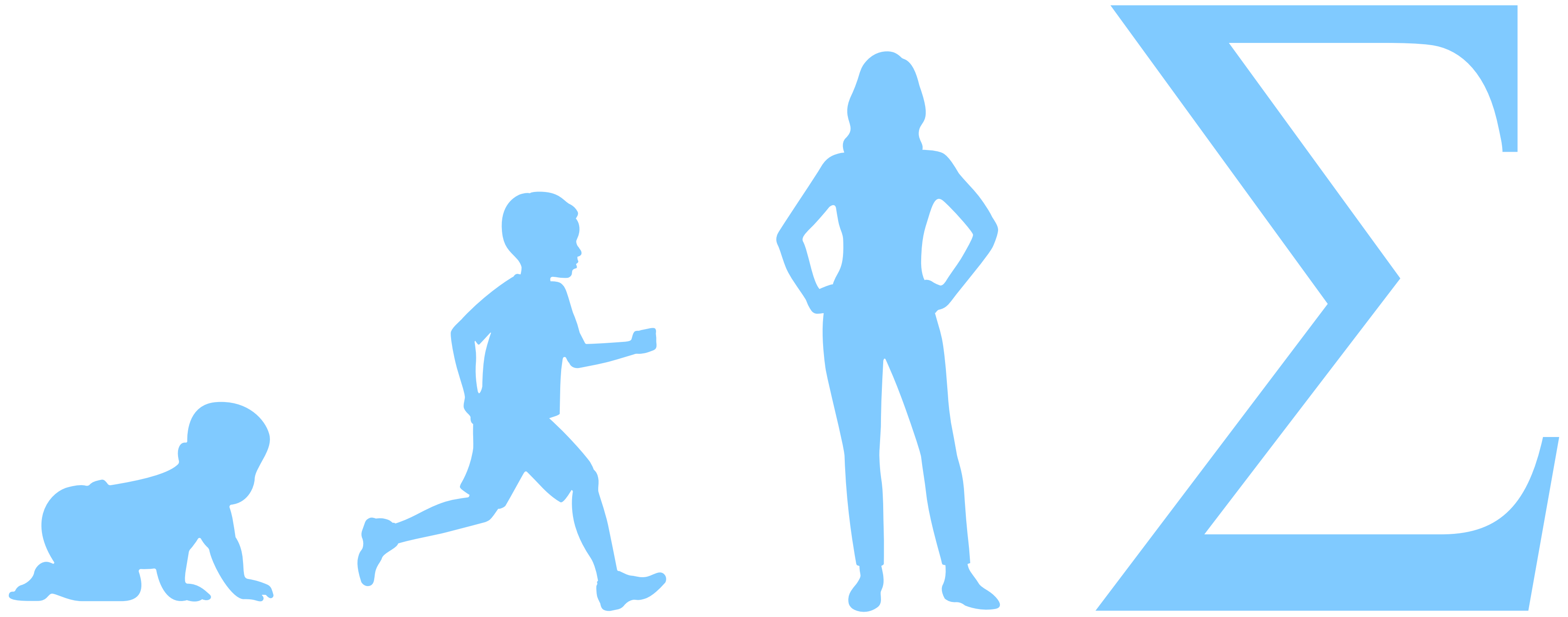
Grading

- **90% presentations** (rubric online)
- **10% in-class participation**

Format Of Class

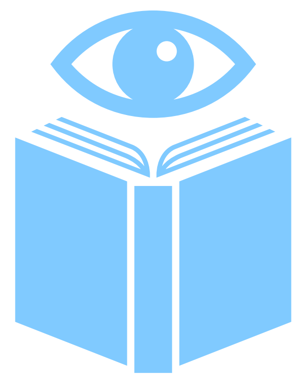
Disclaimer: A Theory Class

- All proof-based; very technical papers
- Pre-reqs:
  - Only official: 155 or 157
  - Mathematical maturity
  - Familiarity with (graph) algorithms useful
  - Relevant background for papers on website
- Ask me if not sure about pre-reqs

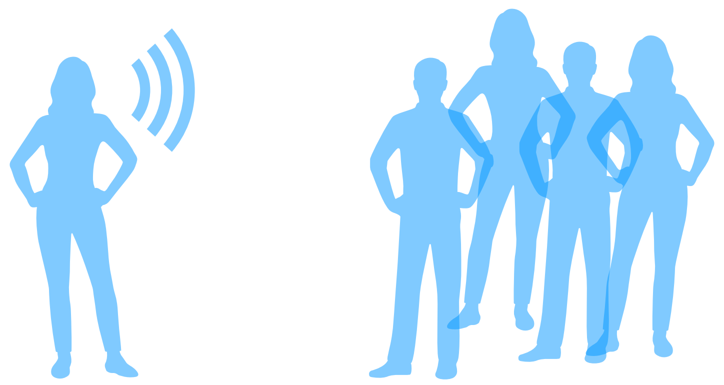


Learning Goals

- Aimed at current / possible (theory) grad students
- Experience with:
  - Reading theory (research papers)



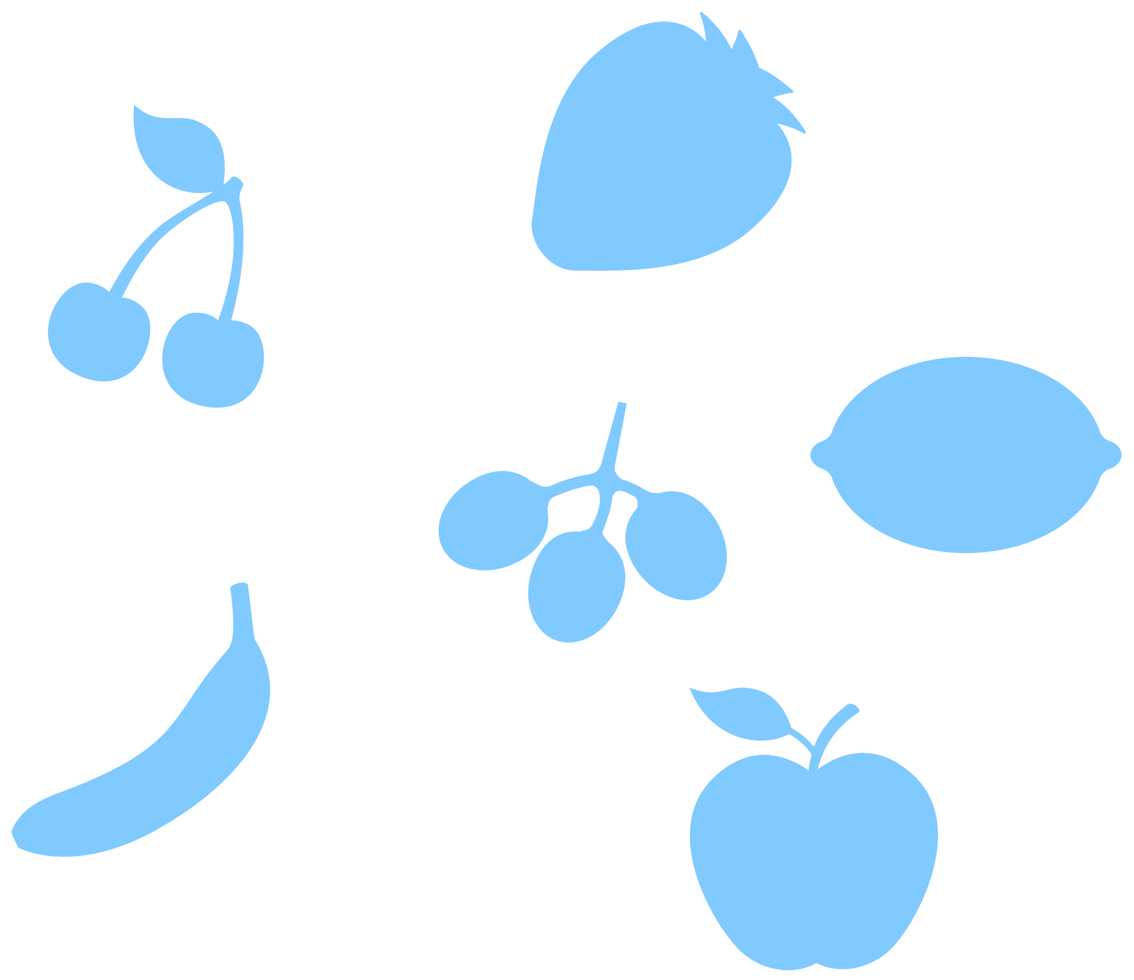
  - Presenting theory (research papers)



  - Listening to theory (research)

Snacks

- I’m planning on bringing snacks (of fruit variety)
- Let me know if you have allergies

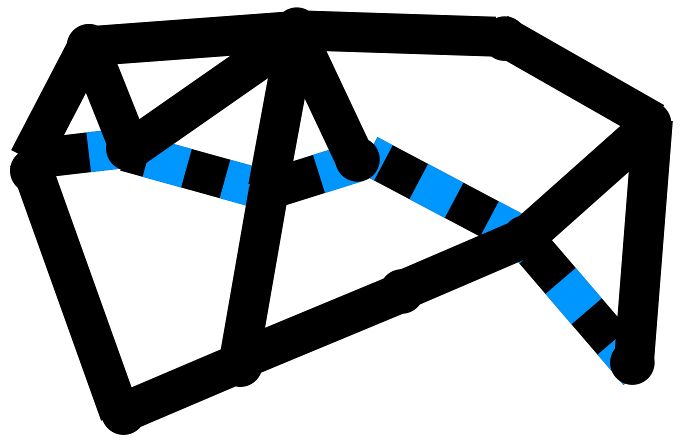


Papers Overview

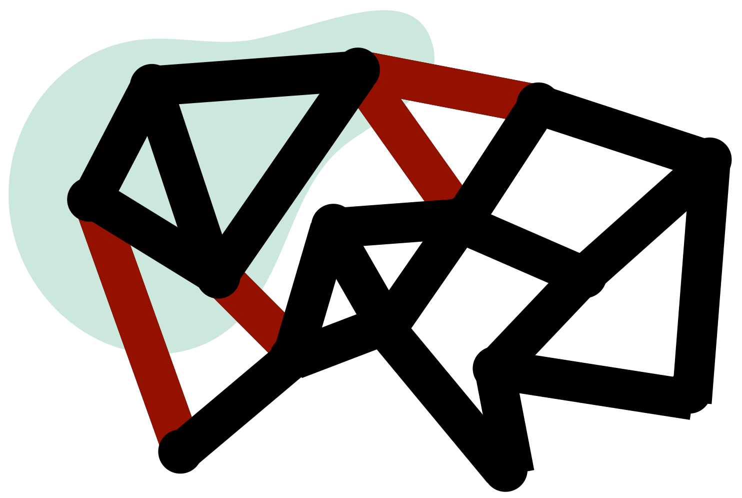
Papers Overview

Sparsification of Five Graph-Theoretic Objects

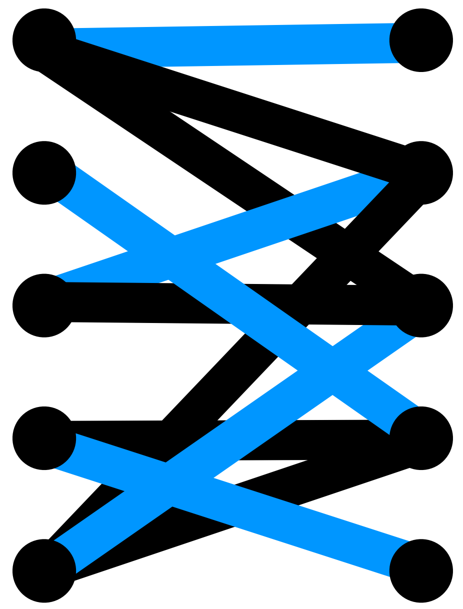
Distances



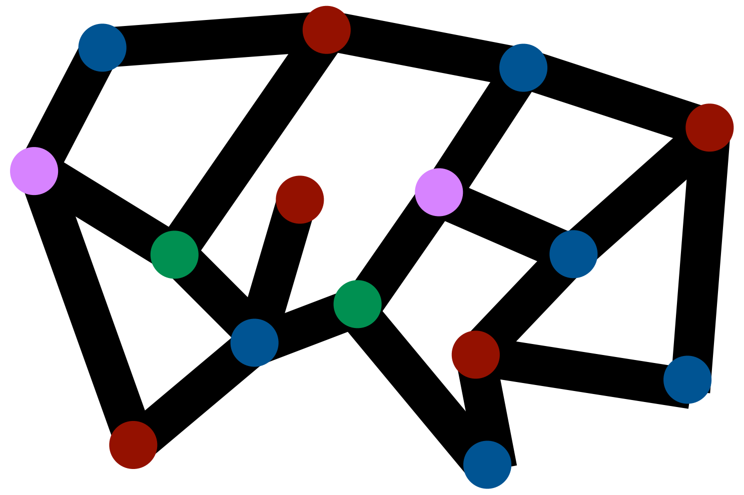
Cuts/Flows



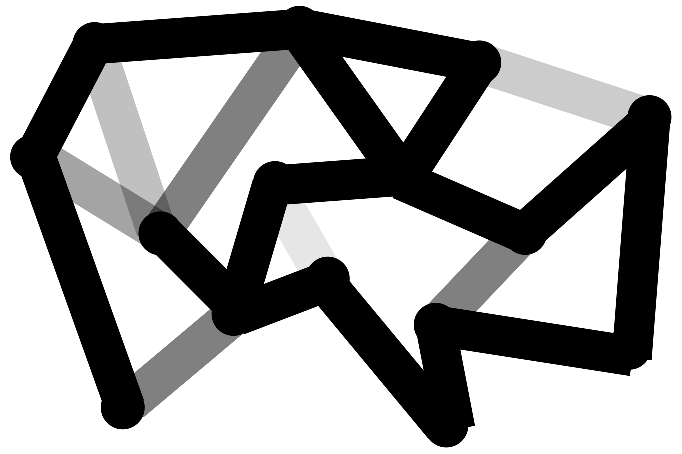
Matchings



Colorings

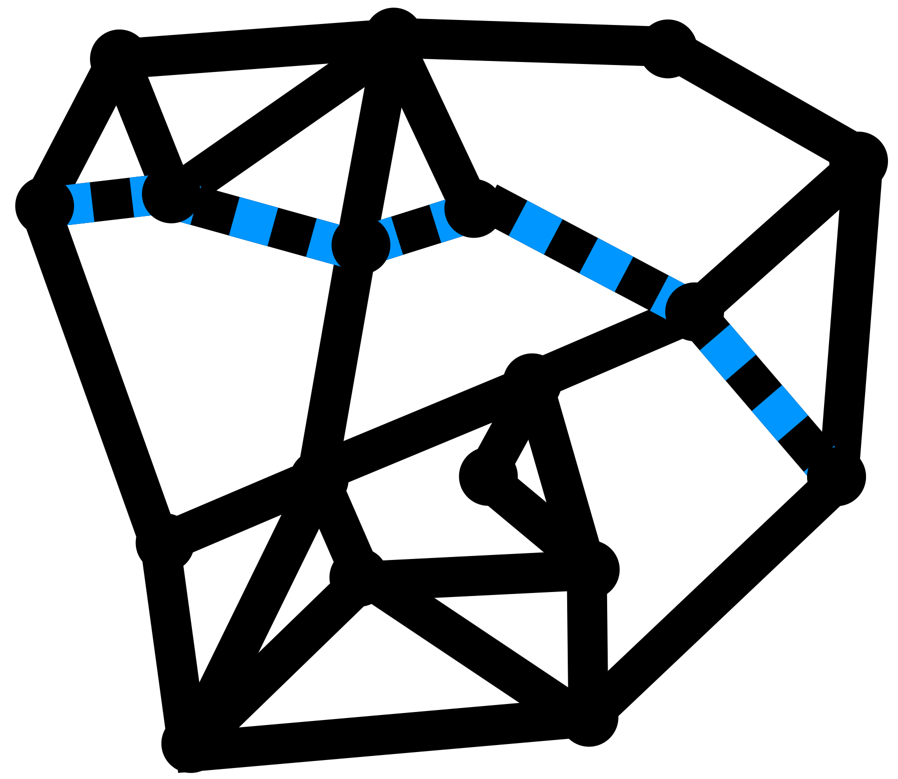


Fractional Opts



Papers Overview

Distance Sparsification



graph $G = (V, E)$

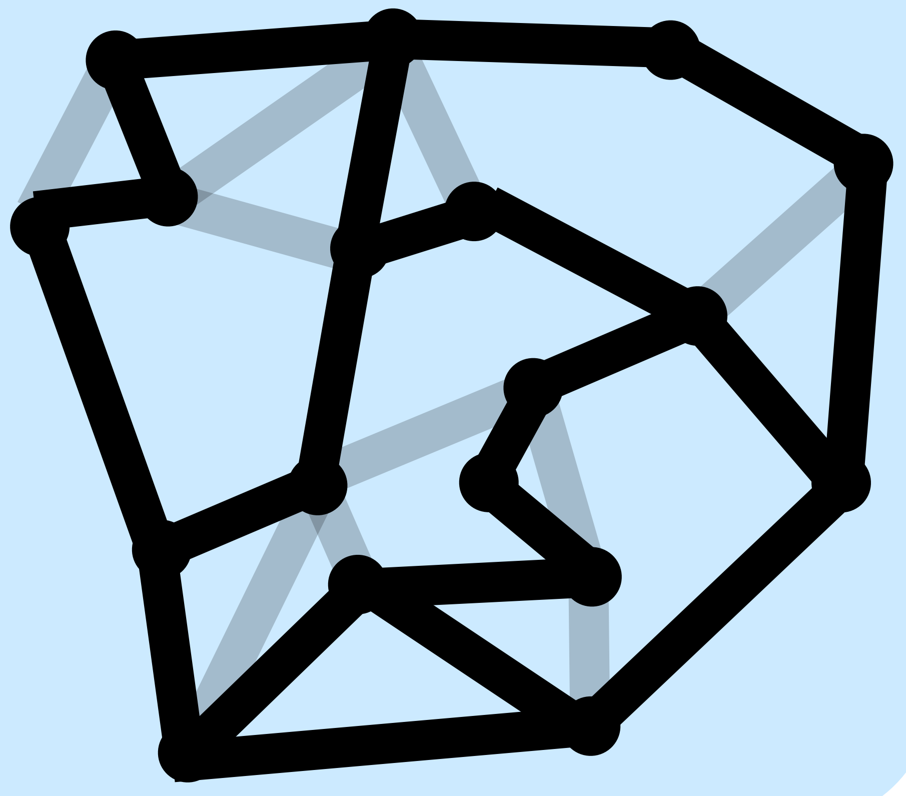
1

Edge sparsification

graph $H = (V, E' \subseteq E)$

$d_H \approx d_G$

(spanners)



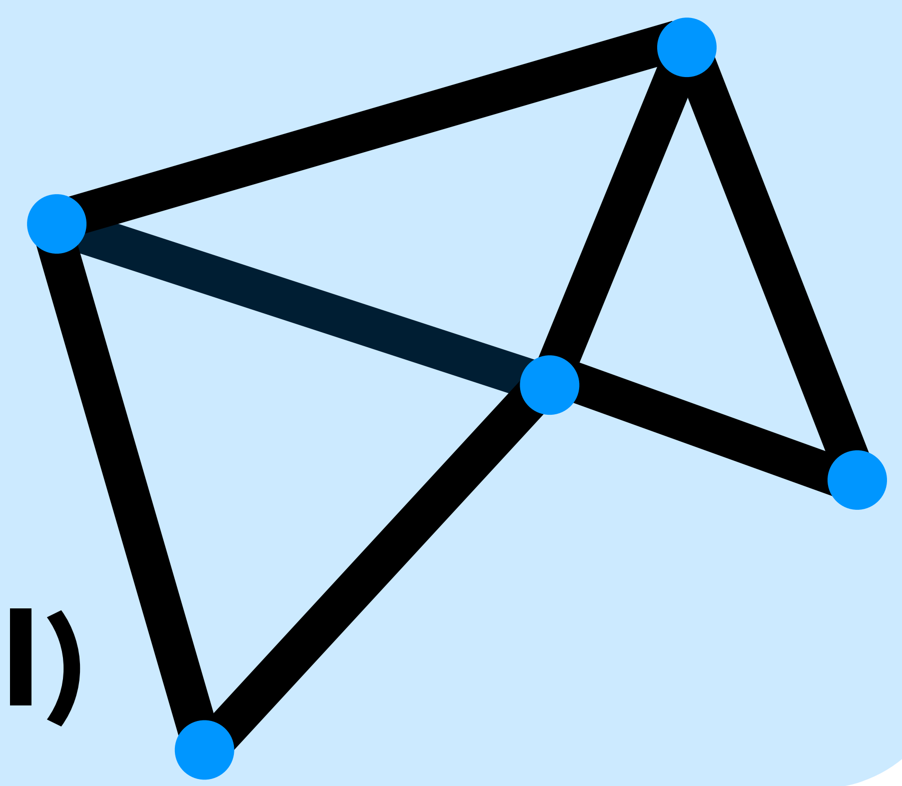
2

Node sparsification

graph $H = (V' \subseteq V, E')$

$d_H \approx d_G$ on V'

(Steiner Point Removal)



3

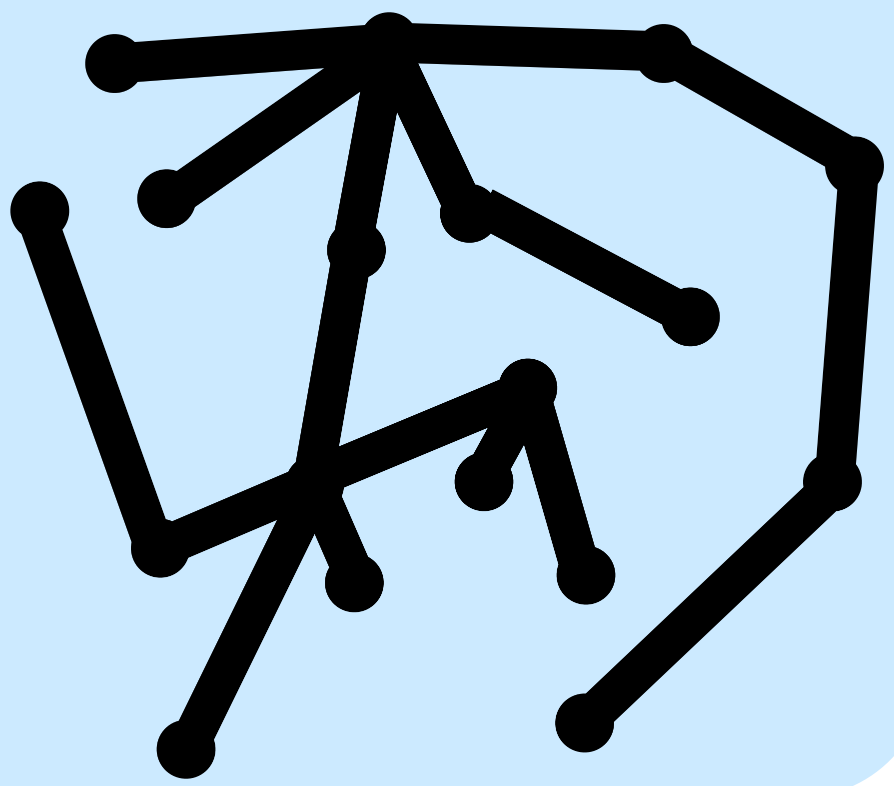
4

Structure sparsification

random tree $T = (V, E')$

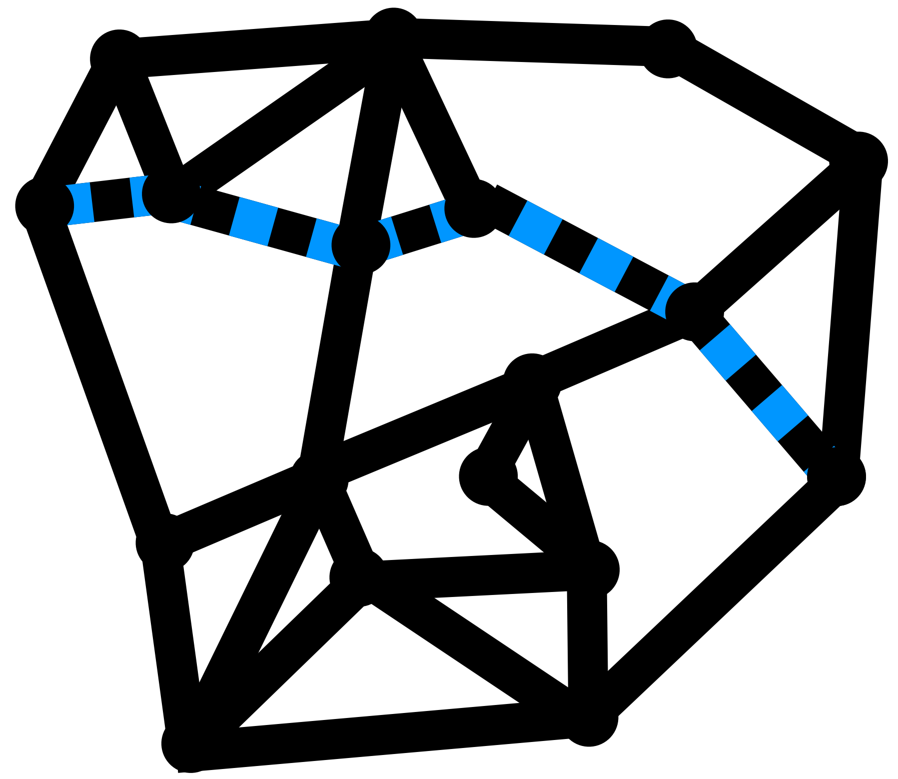
$\mathbb{E}[d_T] \approx d_G$

(Tree Embeddings)



Papers Overview

Distance Sparsification



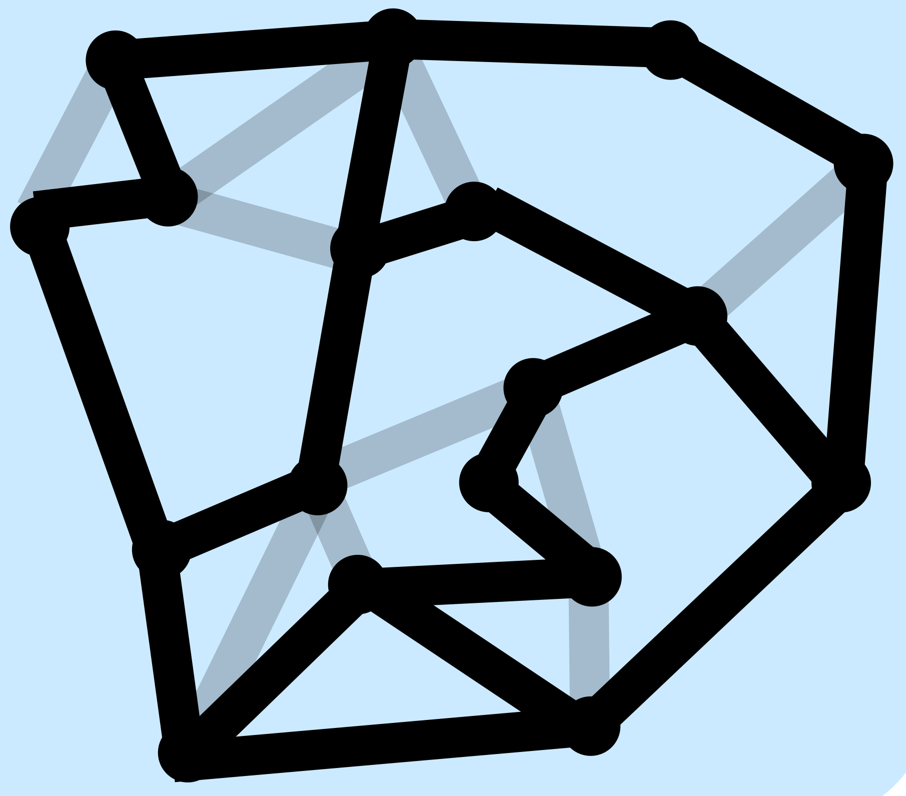
graph $G = (V, E)$

1 ✓ Edge sparsification

graph $H = (V, E' \subseteq E)$

$d_H \approx d_G$

(spanners)

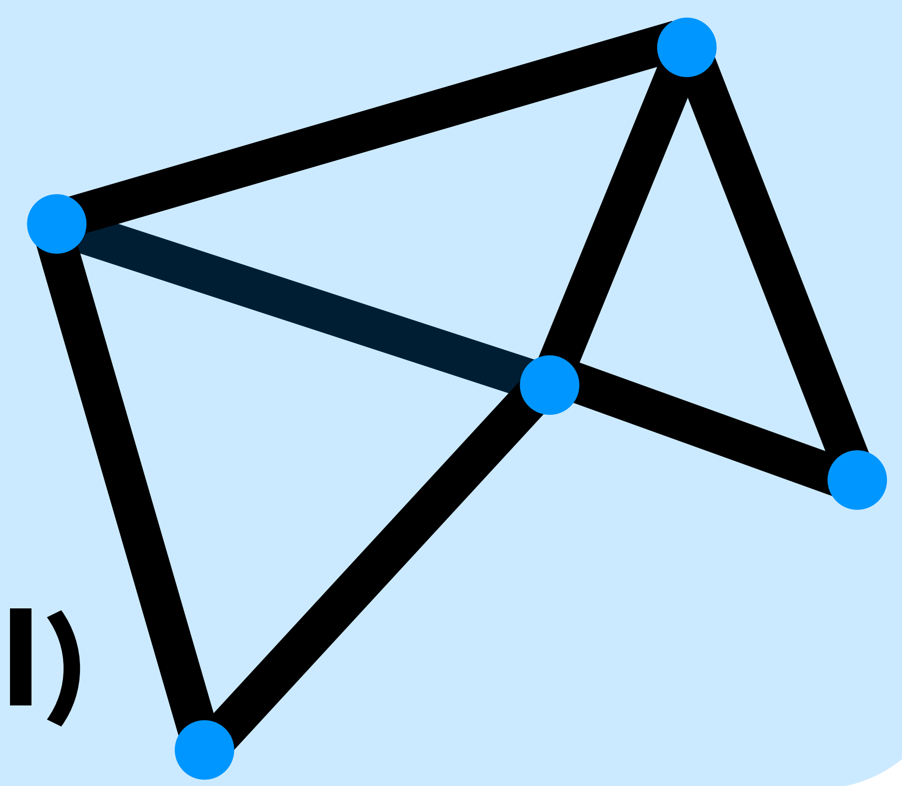


2 Node sparsification

graph $H = (V' \subseteq V, E')$

$d_H \approx d_G$ on V'

(Steiner Point Removal)

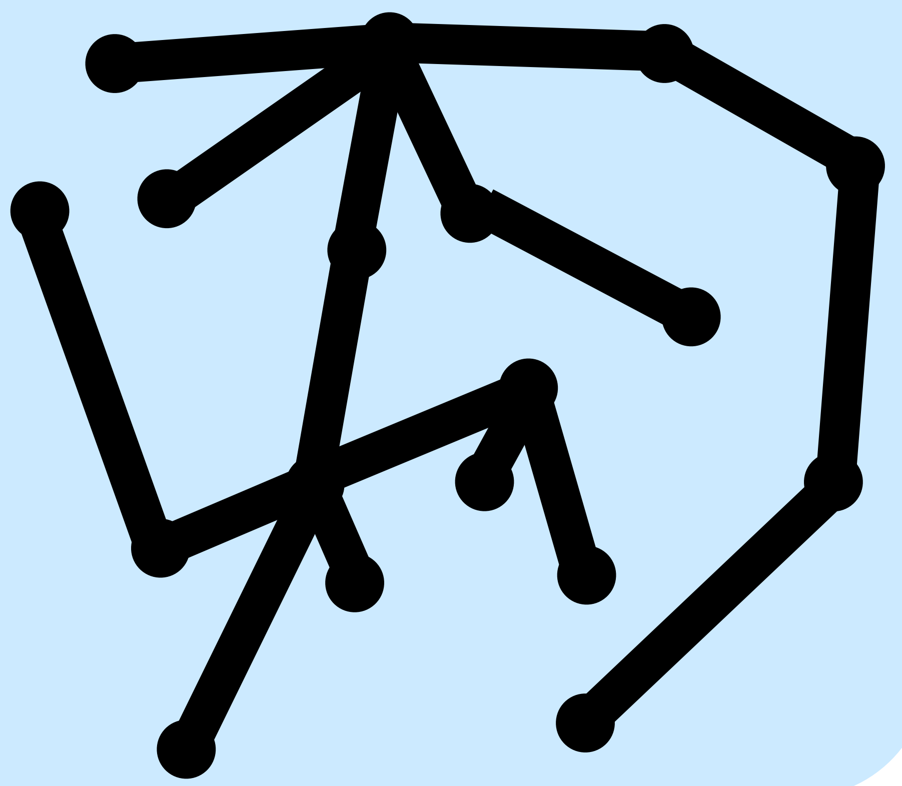


3 **4** Structure sparsification

random tree $T = (V, E')$

$\mathbb{E}[d_T] \approx d_G$

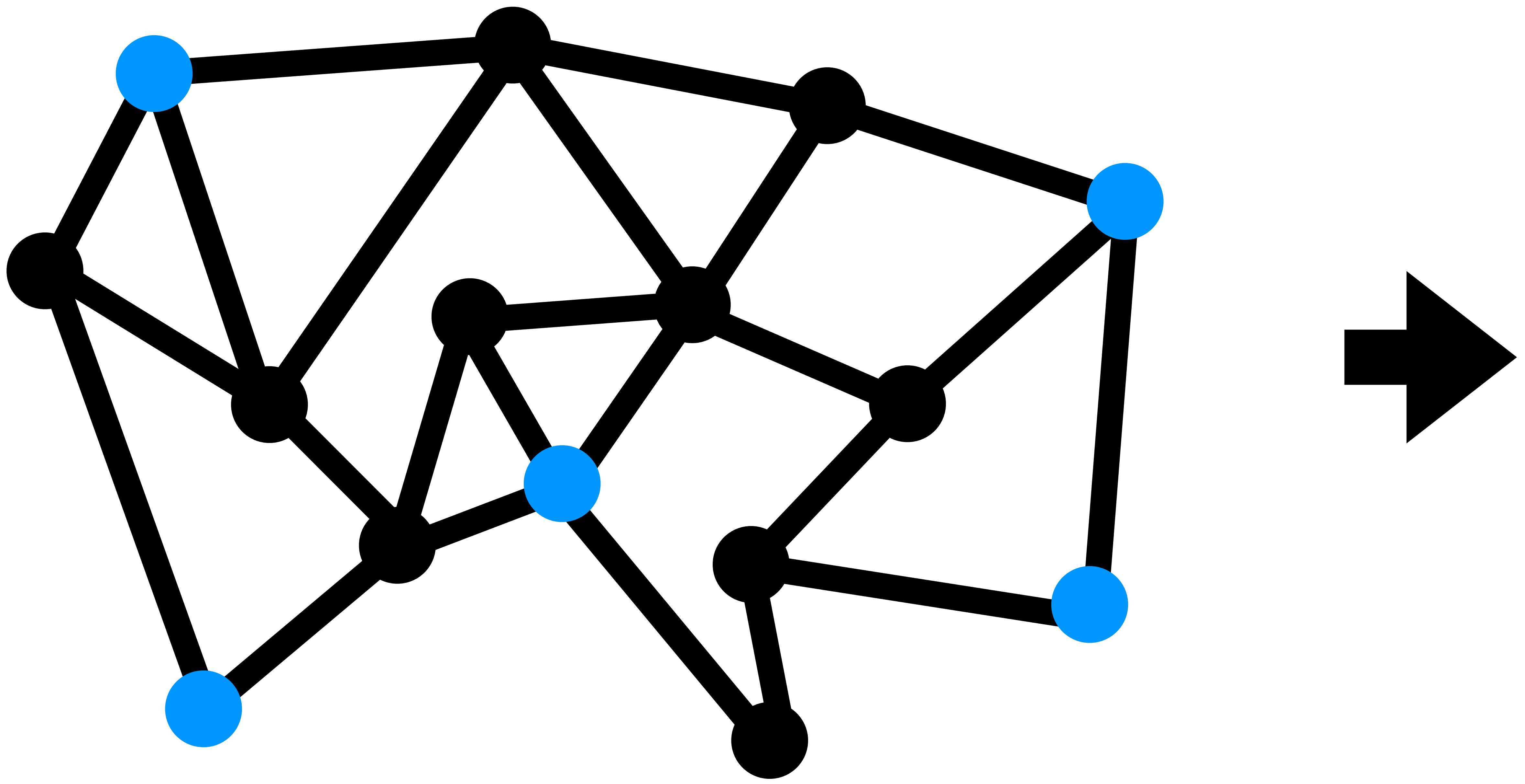
(Tree Embeddings)



Papers Overview

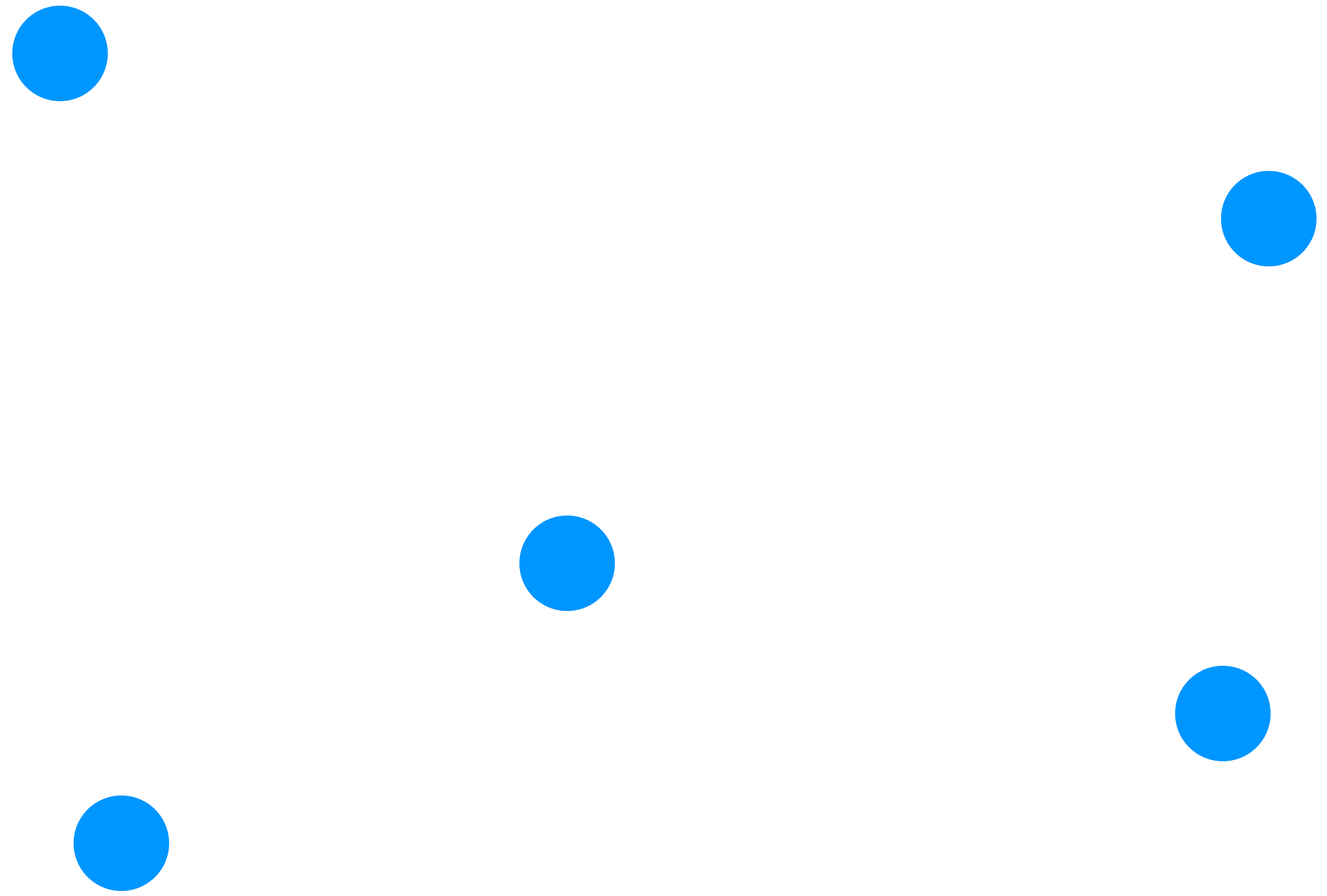
Paper 2: Steiner Point Removal

Trivial as stated!



graph $G = (V, E)$

terminals $T \subseteq V$



graph $H = (T, E', w)$

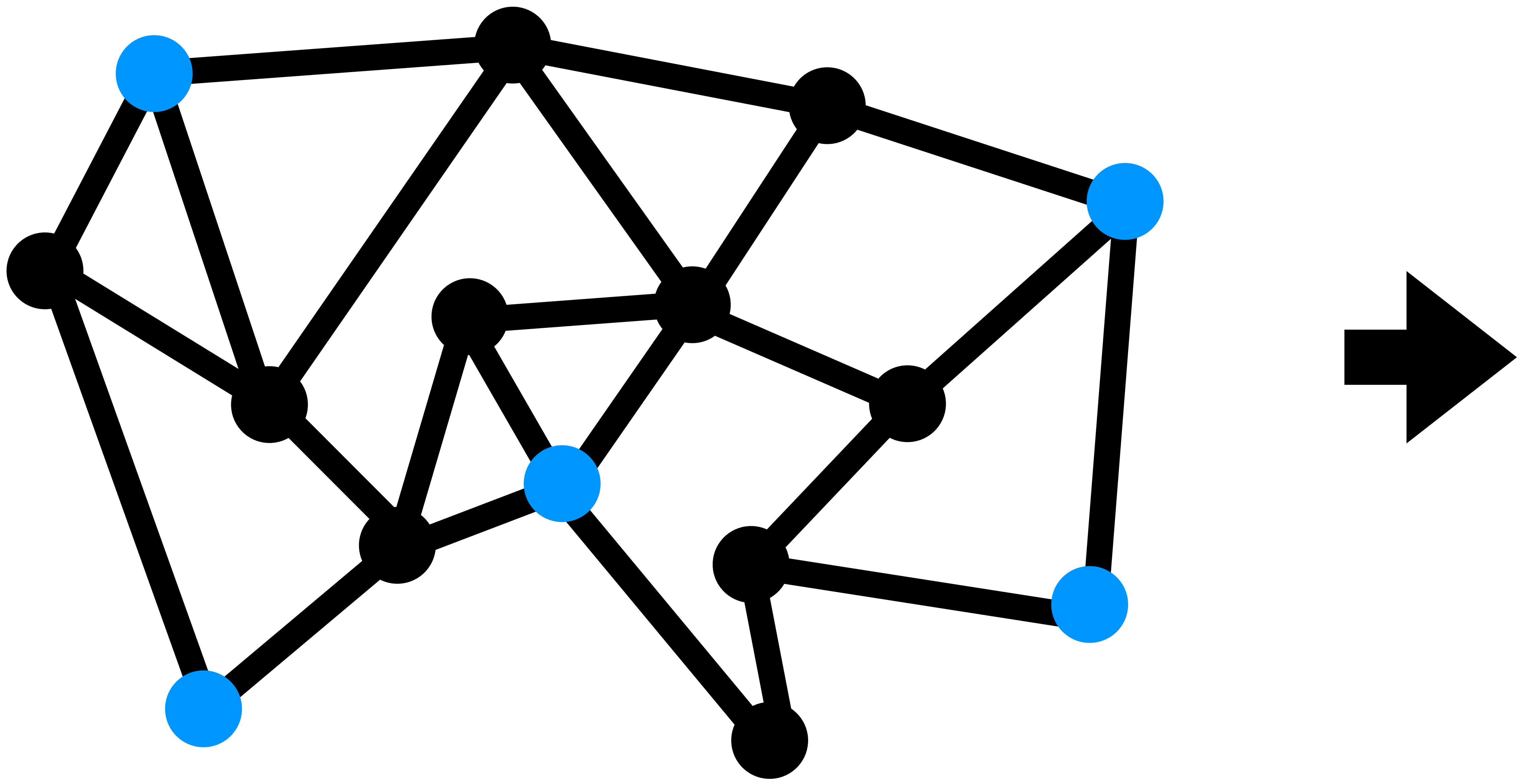
s.t. $d_H \approx d_G$ on T

Goal: approximate distances on vertex subset

Papers Overview

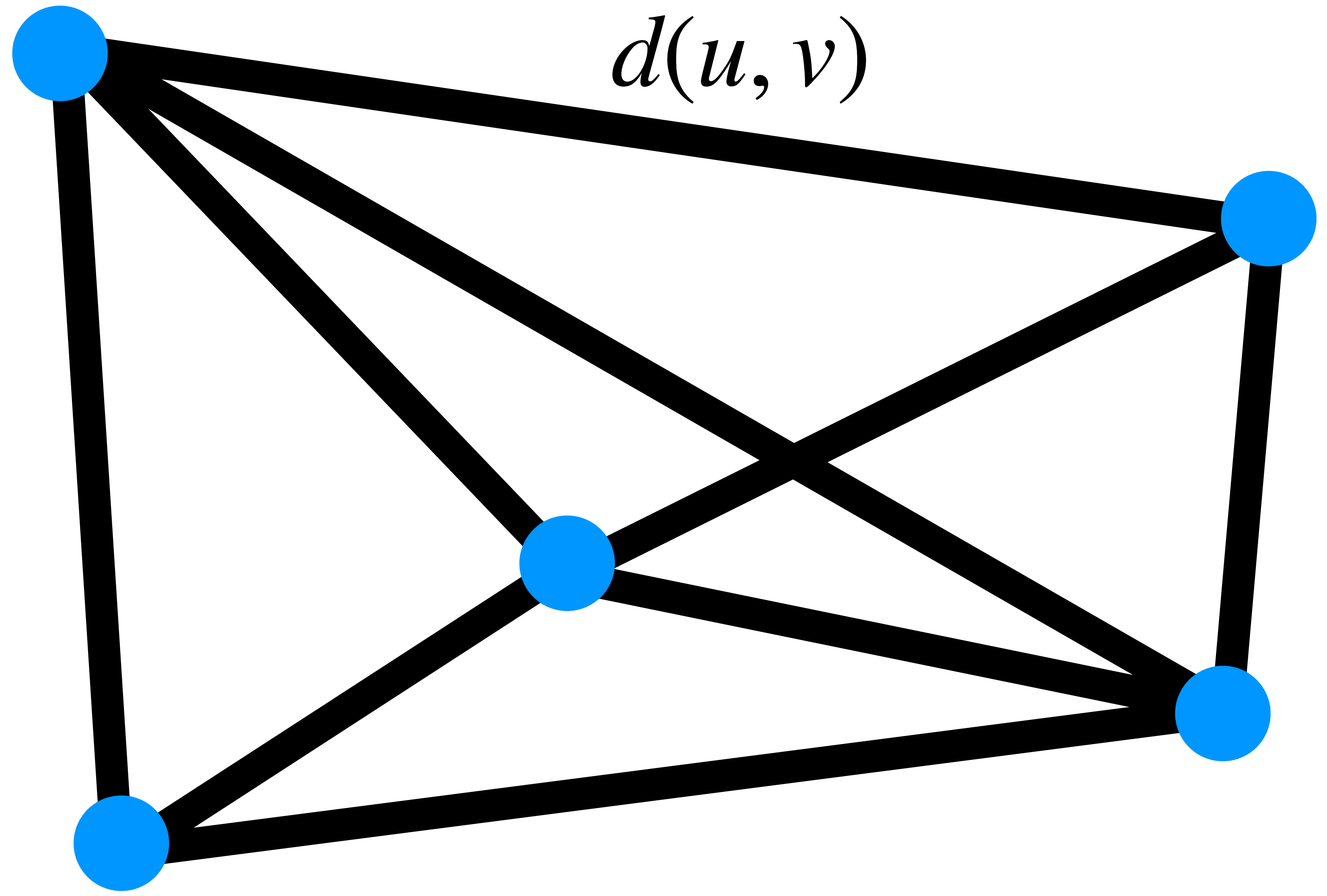
Paper 2: Steiner Point Removal

Trivial as stated!



graph $G = (V, E)$

terminals $T \subseteq V$



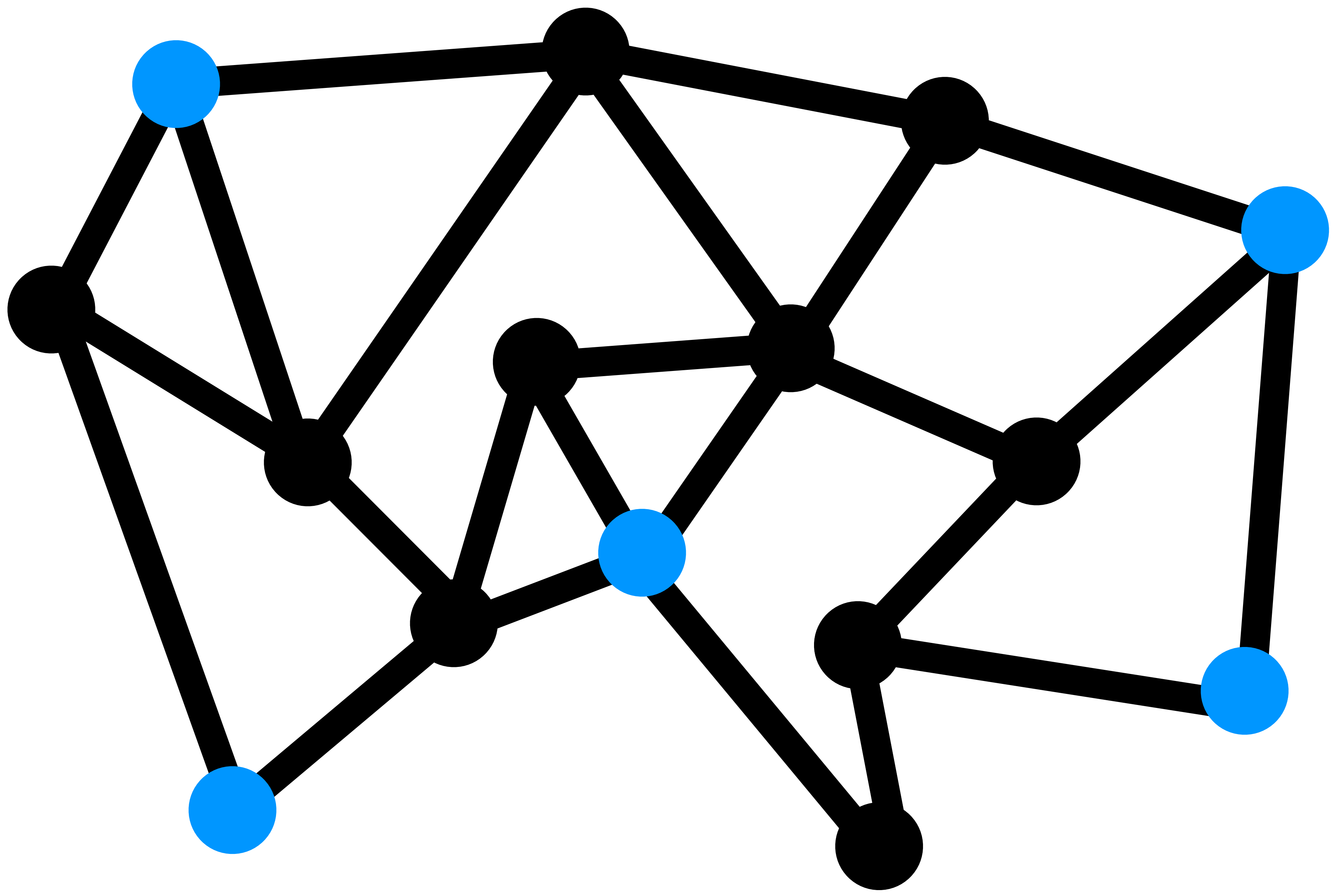
graph $H = (T, E', w)$

s.t. $d_H \approx d_G$ on T

Goal: approximate distances on vertex subset

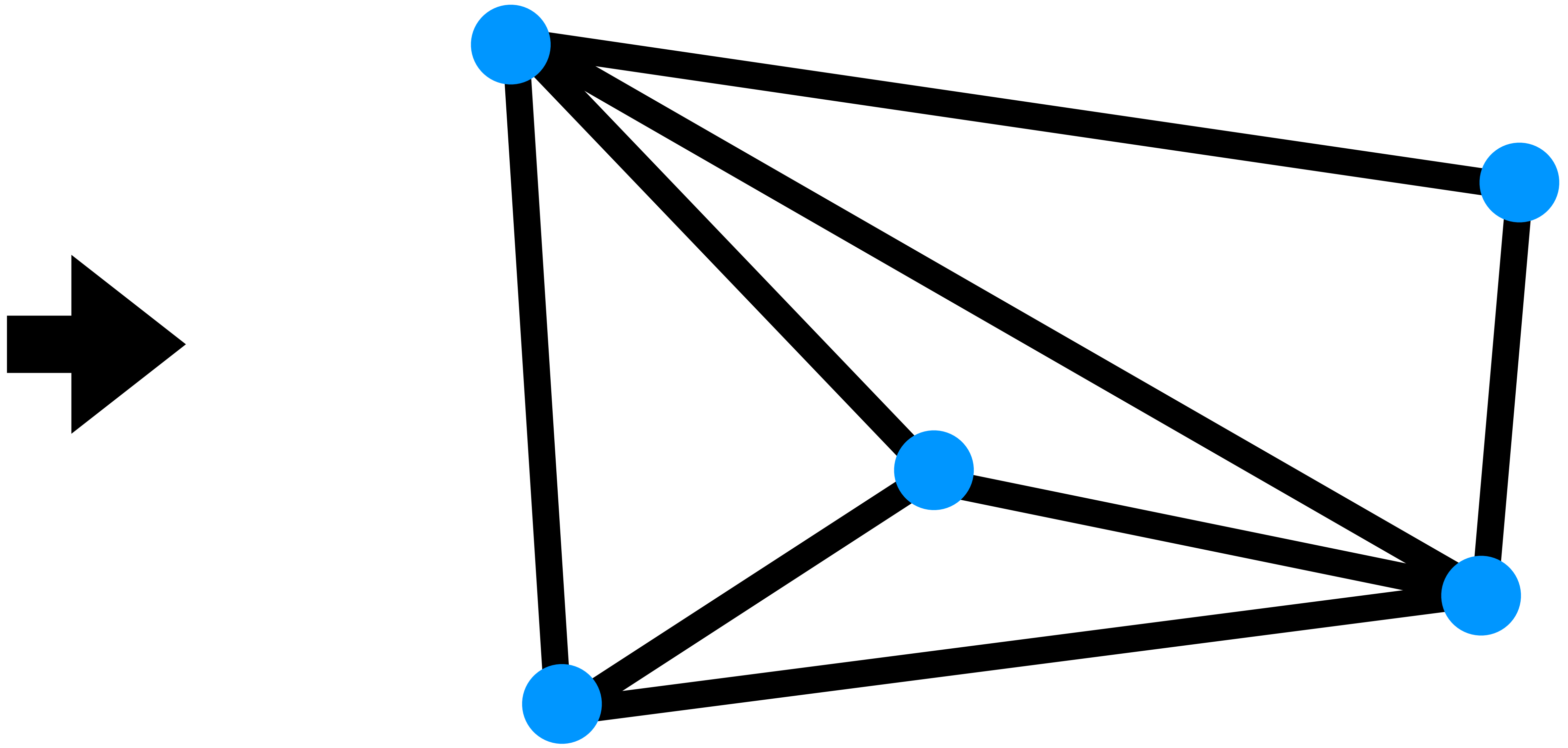
Papers Overview

Paper 2: Steiner Point Removal



graph $G = (V, E)$

terminals $T \subseteq V$



graph $H = (T, E', w)$

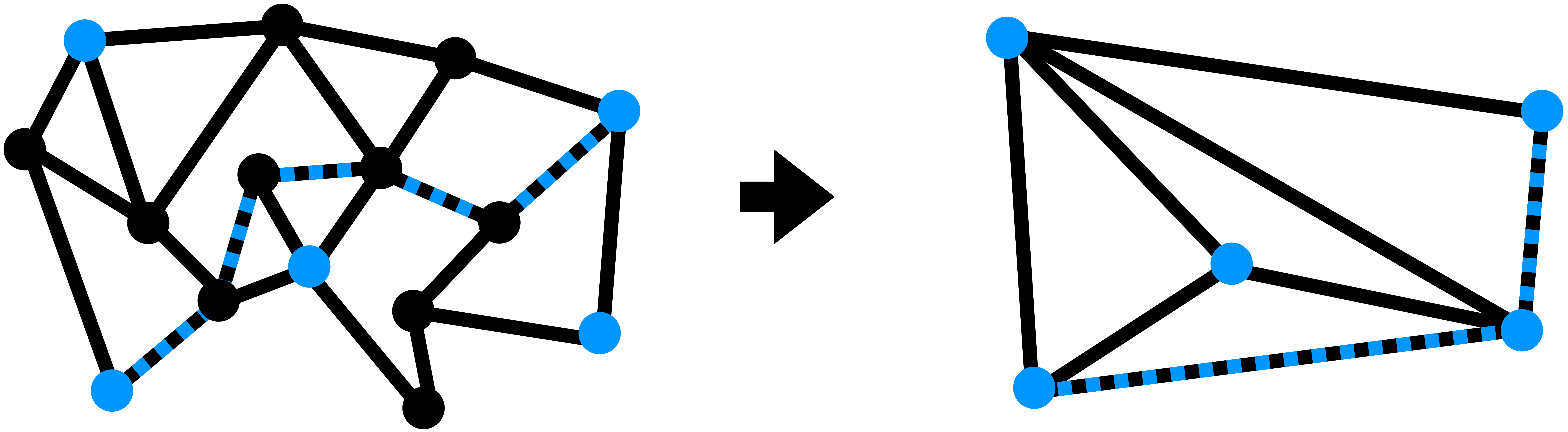
that preserves G's ``structure''

(i.e. is a minor)

s.t. $d_H \approx d_G$ on T

Papers Overview

Paper 2: Steiner Point Removal

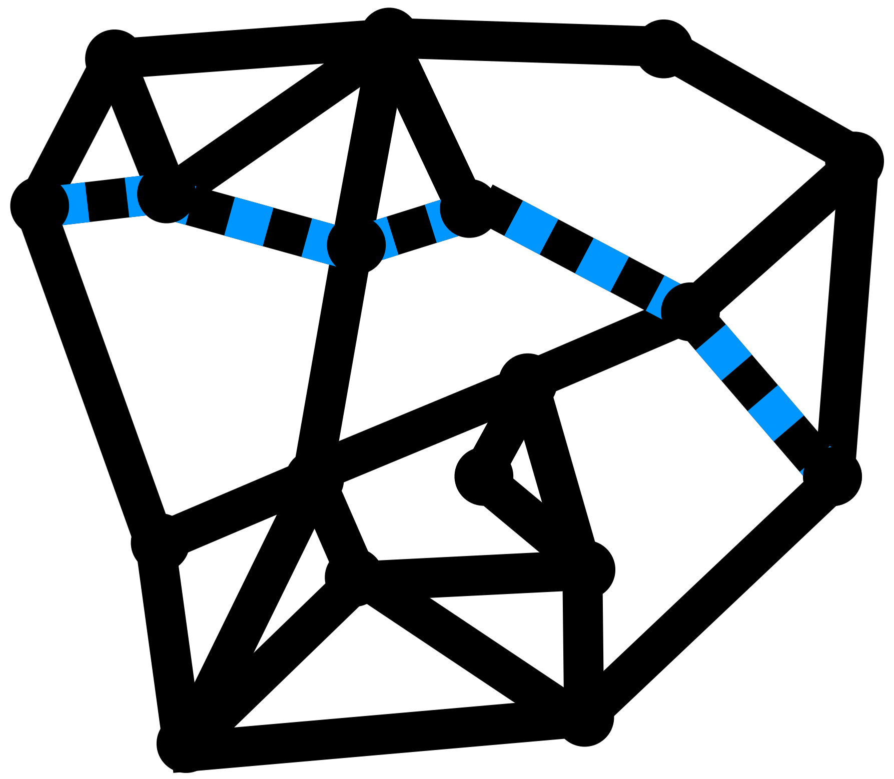


Theorem: given $G = (V, E)$ and $T \subseteq V$, there is an edge-weighted minor H s.t.

$$d_G(u, v) \leq d_H(u, v) \leq O(\log |T|) \cdot d_G(u, v) \quad \forall u, v \in T$$

Papers Overview

Distance Sparsification



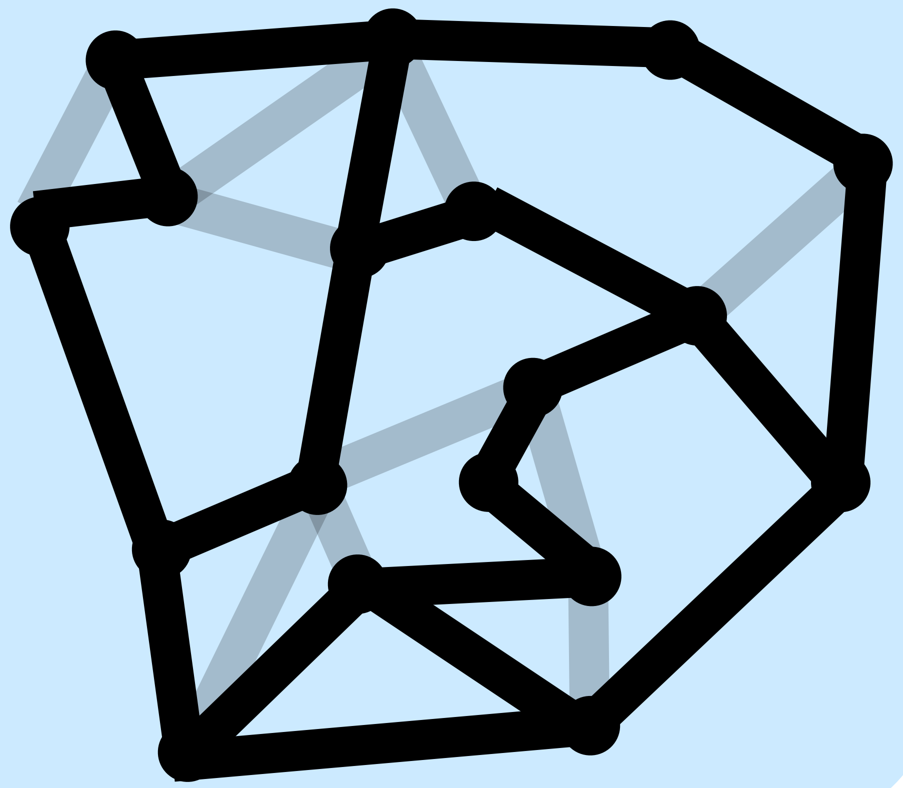
graph $G = (V, E)$

1 ✓ Edge sparsification

graph $H = (V, E' \subseteq E)$

$d_H \approx d_G$

(spanners)

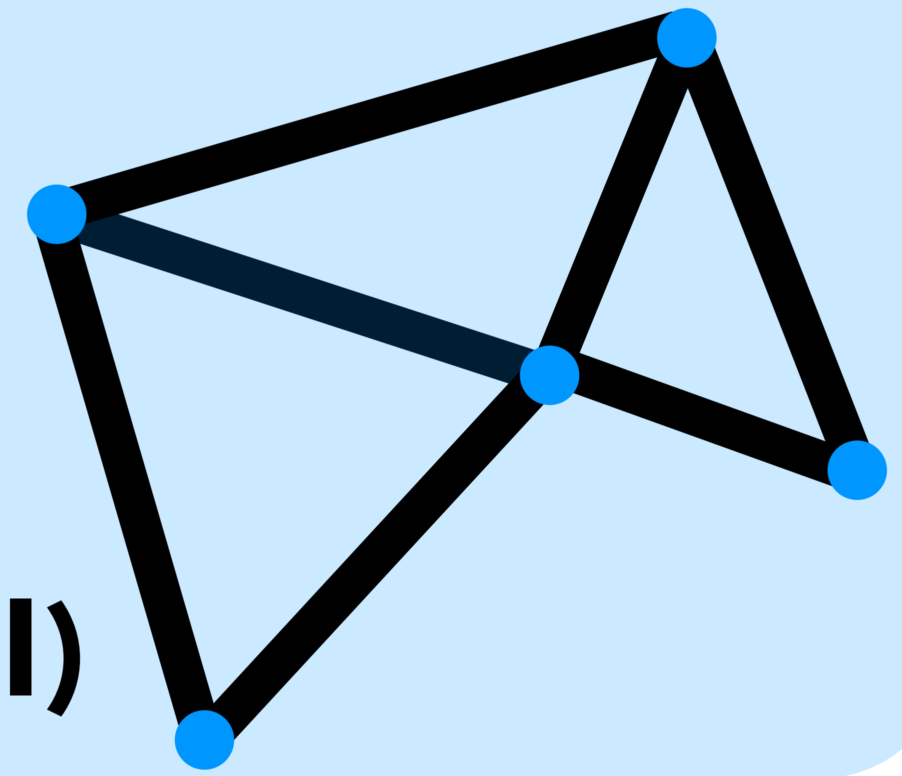


2 Node sparsification

graph $H = (V' \subseteq V, E')$

$d_H \approx d_G$ on V'

(Steiner Point Removal)



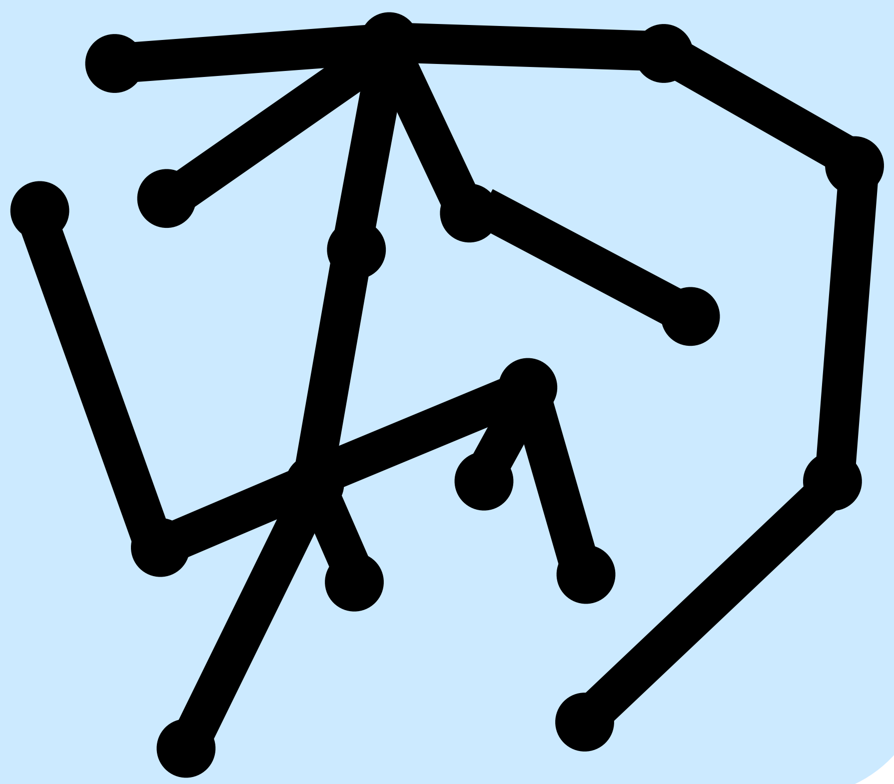
3

4 Structure sparsification

random tree $T = (V, E')$

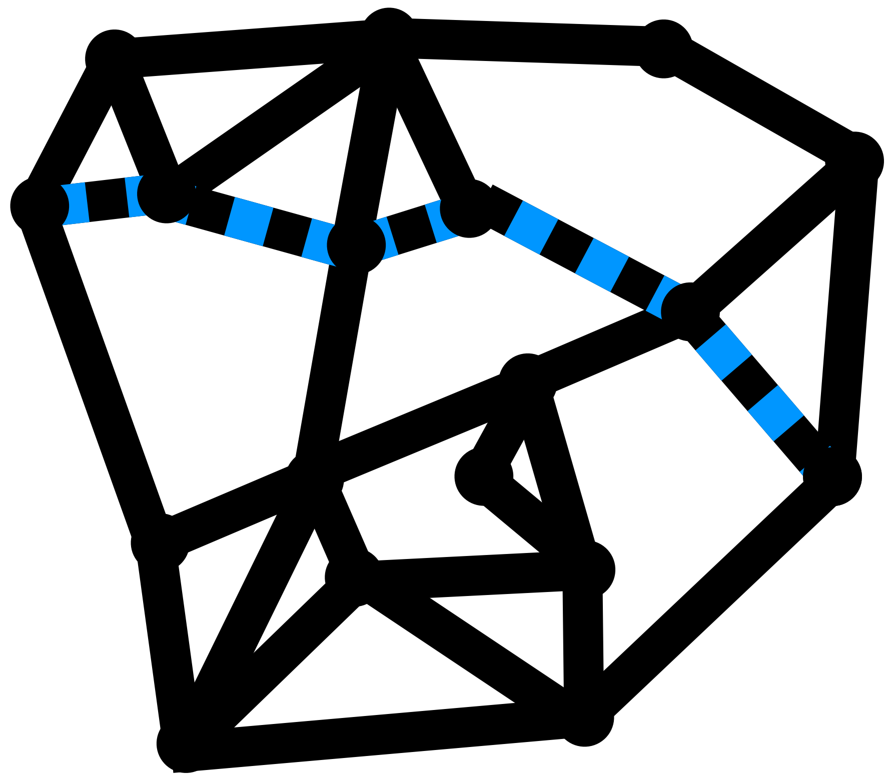
$\mathbb{E}[d_T] \approx d_G$

(Tree Embeddings)



Papers Overview

Distance Sparsification



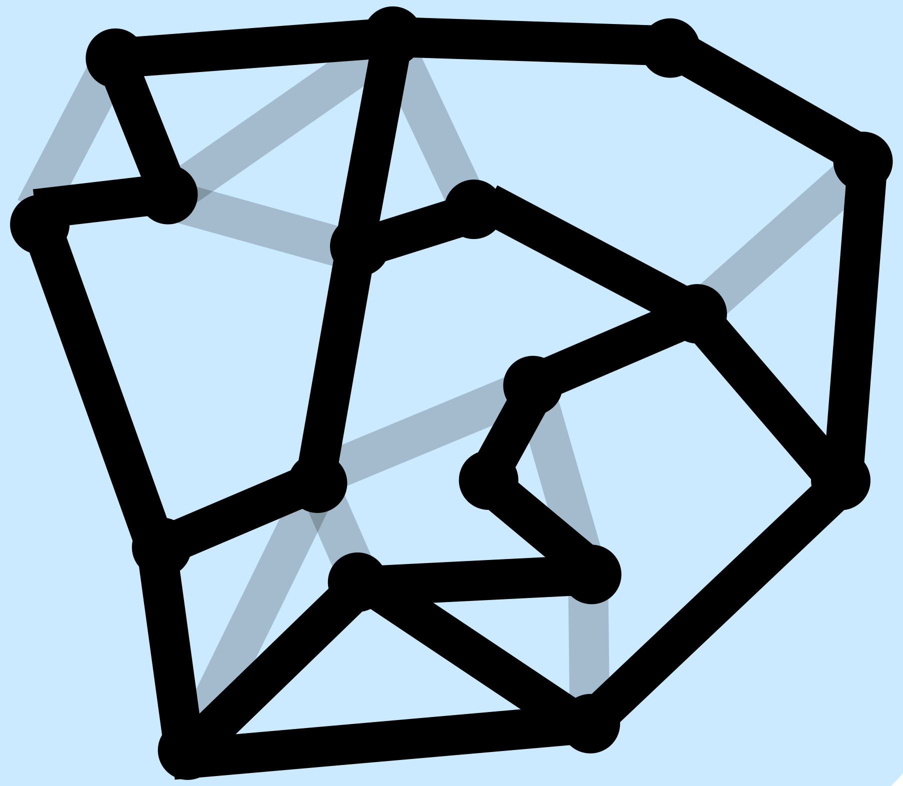
graph $G = (V, E)$

1 ✓ <u>Edge sparsification</u>

graph $H = (V, E' \subseteq E)$

$d_H \approx d_G$

(spanners)

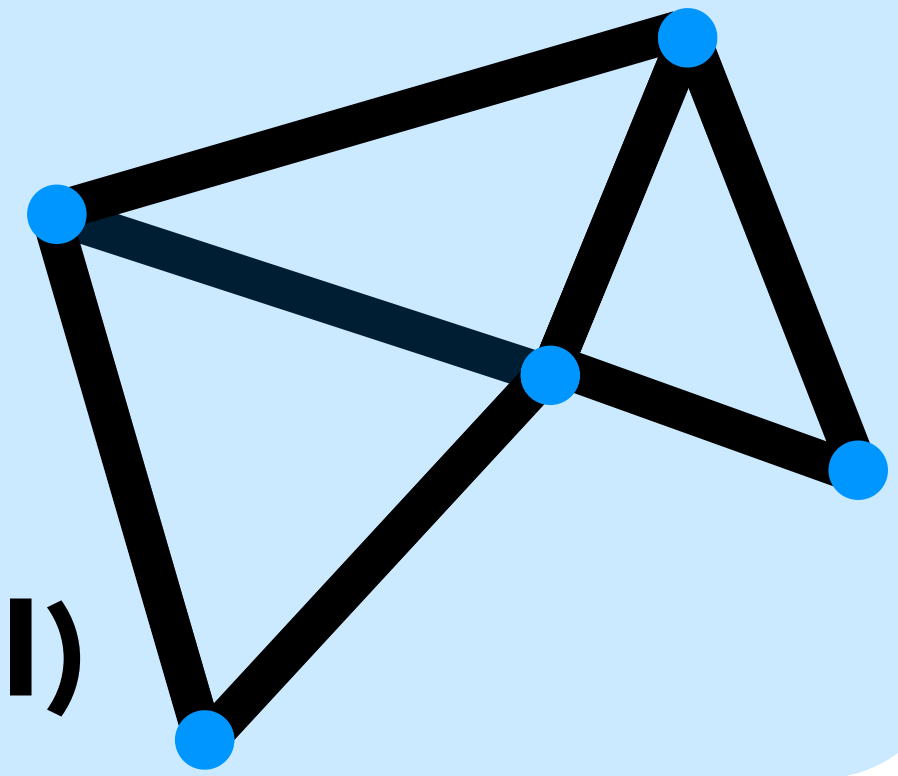


2 ✓ <u>Node sparsification</u>

graph $H = (V' \subseteq V, E')$

$d_H \approx d_G$ on V'

(Steiner Point Removal)



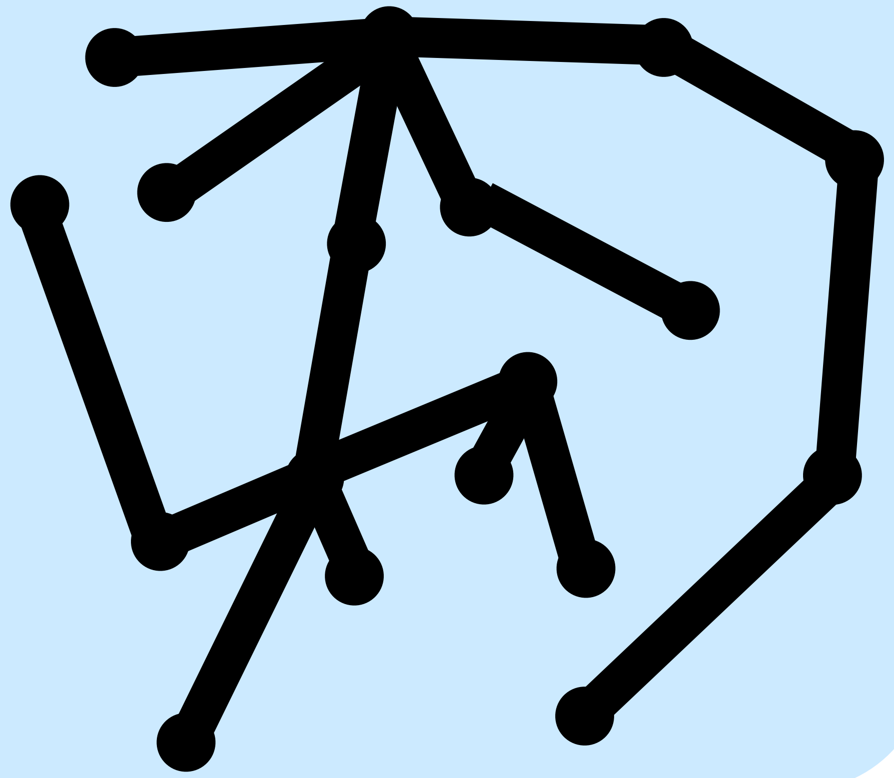
3

4 <u>Structure sparsification</u>

random tree $T = (V, E')$

$\mathbb{E}[d_T] \approx d_G$

(Tree Embeddings)



Papers Overview

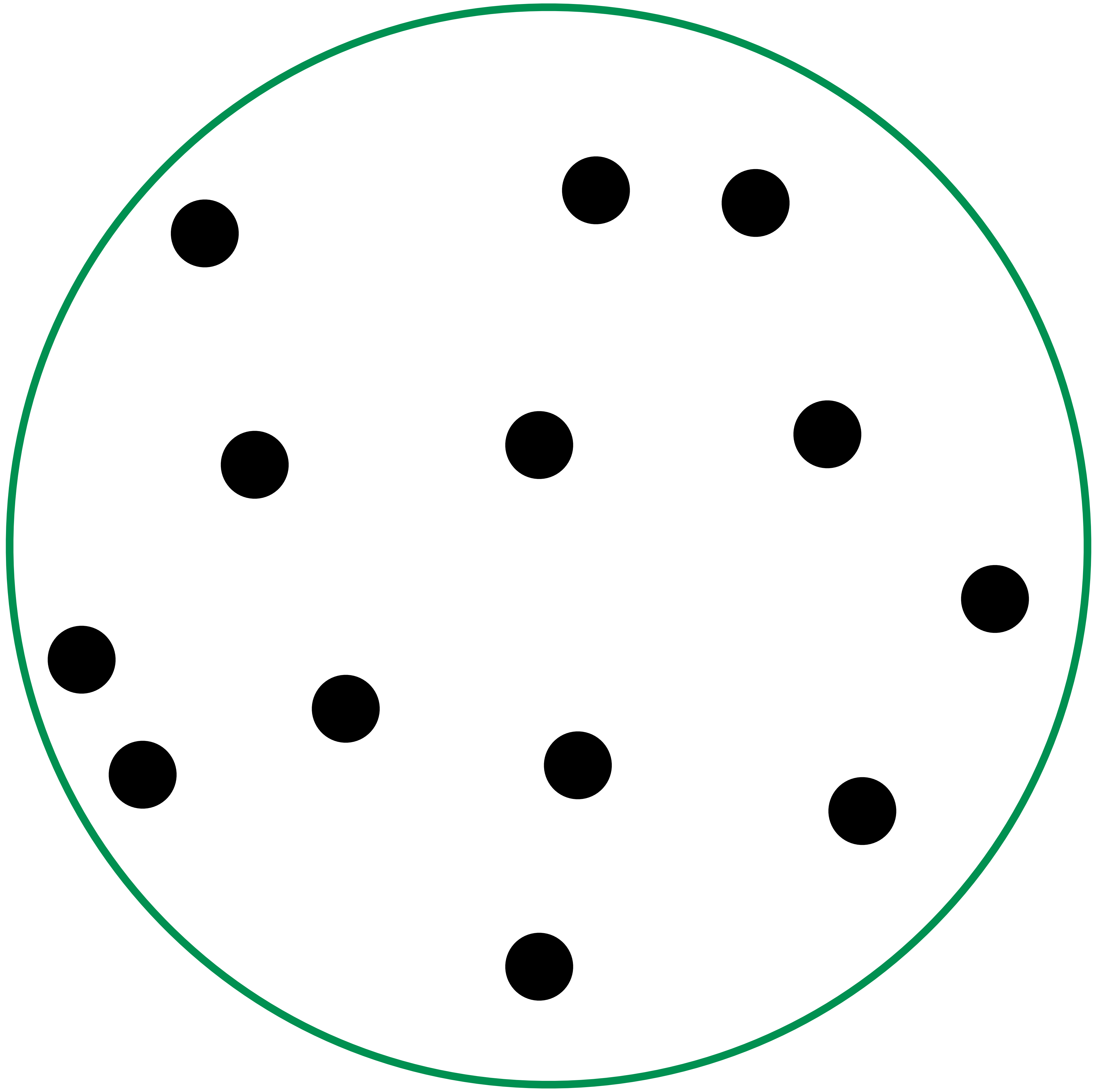
Paper 3: CKR Cutting Scheme

- Partition vertices V into sets $V_1, V_2, \ldots$
- A tradeoff between
  - Low diameter

    $$\max_i \max_{u,v \in V_i} d_G(u, v)$$

  - Low separation

    chances u, v in different V_i



Goal: random low diameter partition with small separation probability

Papers Overview

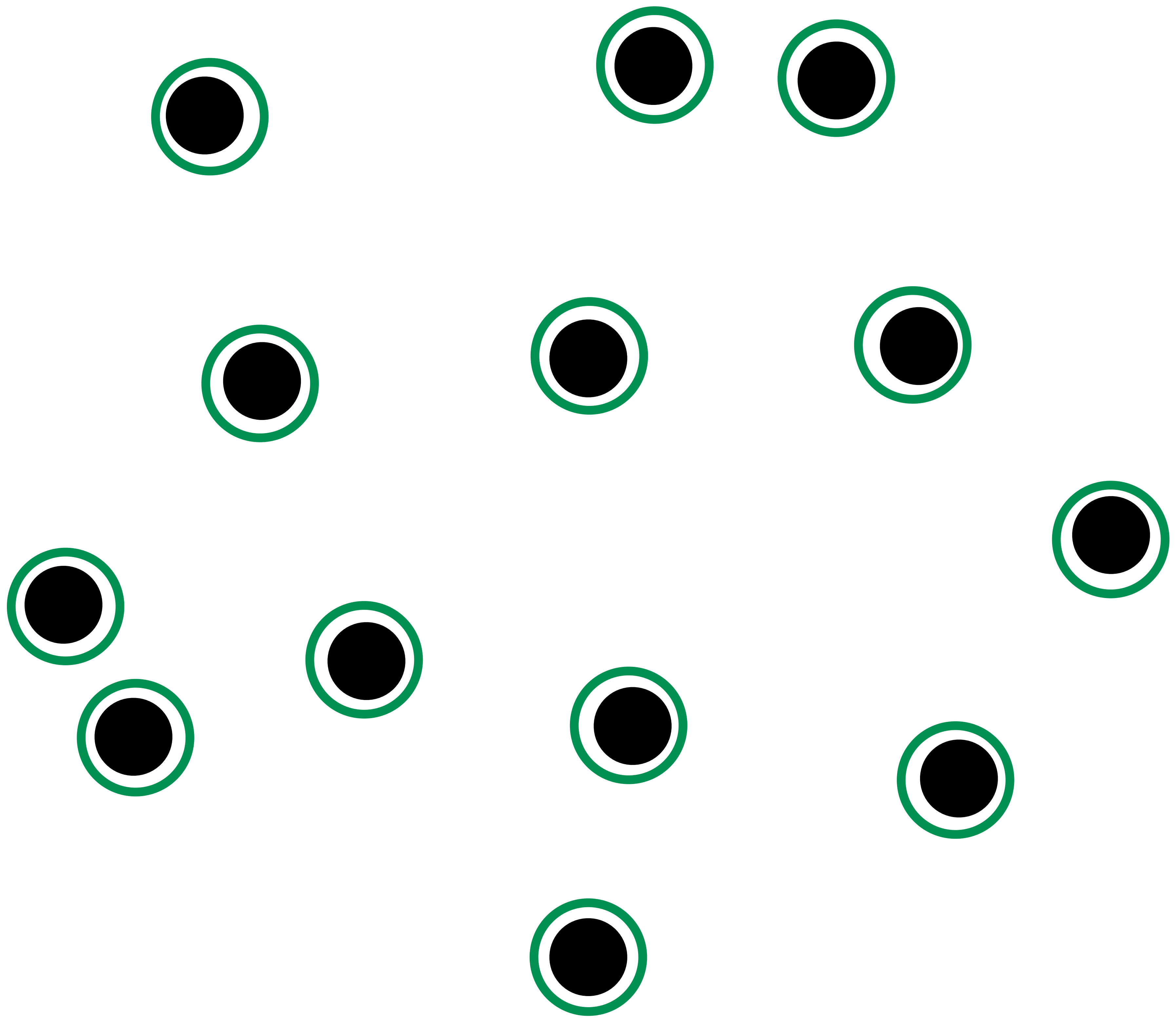
Paper 3: CKR Cutting Scheme

- Partition vertices V into sets $V_1, V_2, \ldots$
- A tradeoff between
  - Low diameter

    $$\max_i \max_{u,v \in V_i} d_G(u, v)$$

  - Low separation

    chances u, v in different V_i



Goal: random low diameter partition with small separation probability

Papers Overview

Paper 3: CKR Cutting Scheme

- Consider partitioning path into Δ-diameter parts
- Randomly shift partition by $U[\Delta]$

Goal: random low diameter partition with small separation probability

Papers Overview

Paper 3: CKR Cutting Scheme

- Consider partitioning path into Δ-diameter parts
- Randomly shift partition by $U[\Delta]$

$$\Pr(u, v \text{ separated}) \leq \frac{d(u, v)}{\Delta} \quad \forall u, v$$

Goal: random low diameter partition with small separation probability

Papers Overview

Paper 3: CKR Cutting Scheme

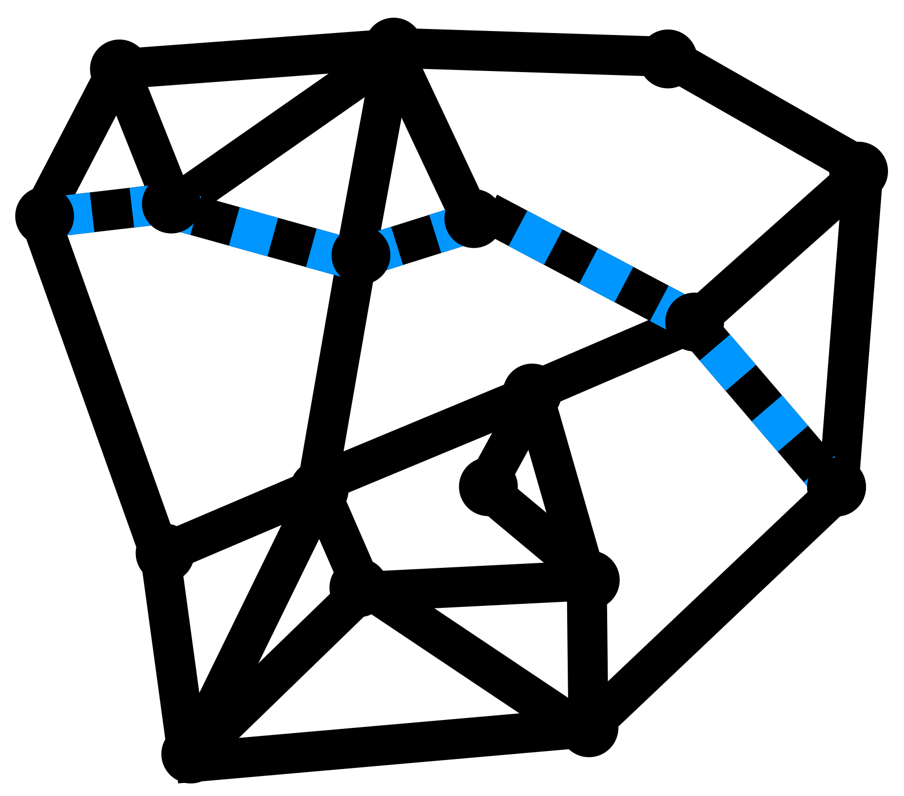
Theorem: given graph G and diameter Δ there exists a distribution over Δ-diameter partitions s.t.

$$\Pr(u, v \text{ separated}) \leq O(\log n) \cdot \frac{d_G(u, v)}{\Delta} \quad \forall u, v \in V$$

(and applications in the ``0-extension'' problem)

Papers Overview

Distance Sparsification



graph $G = (V, E)$

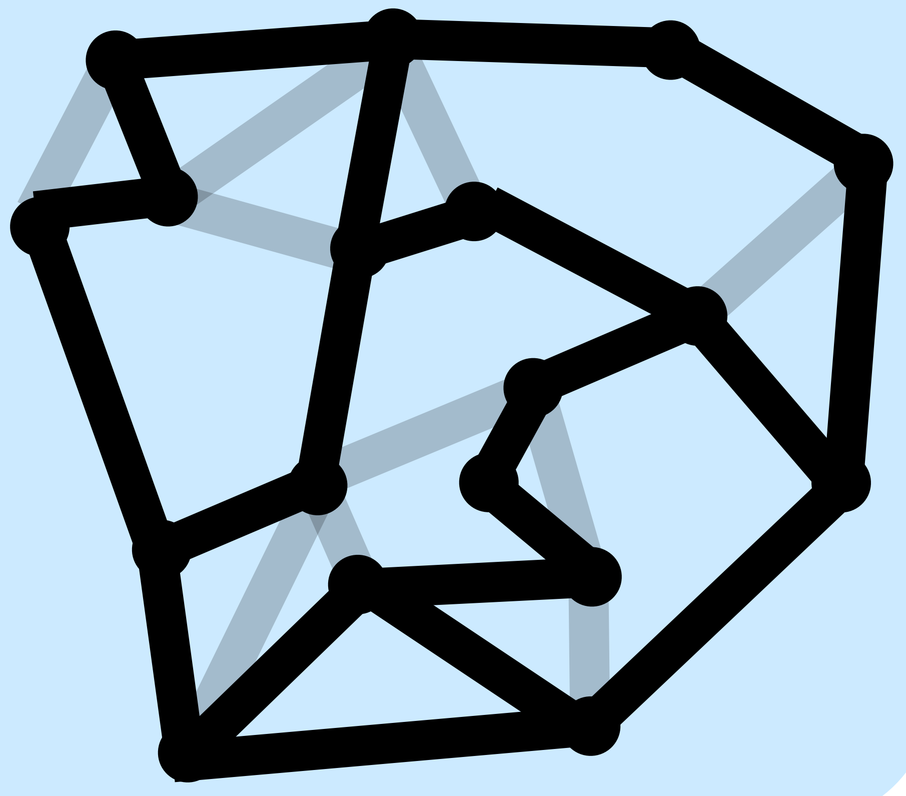
3

1 ✓ <u>Edge sparsification</u>

graph $H = (V, E' \subseteq E)$

$d_H \approx d_G$

(spanners)

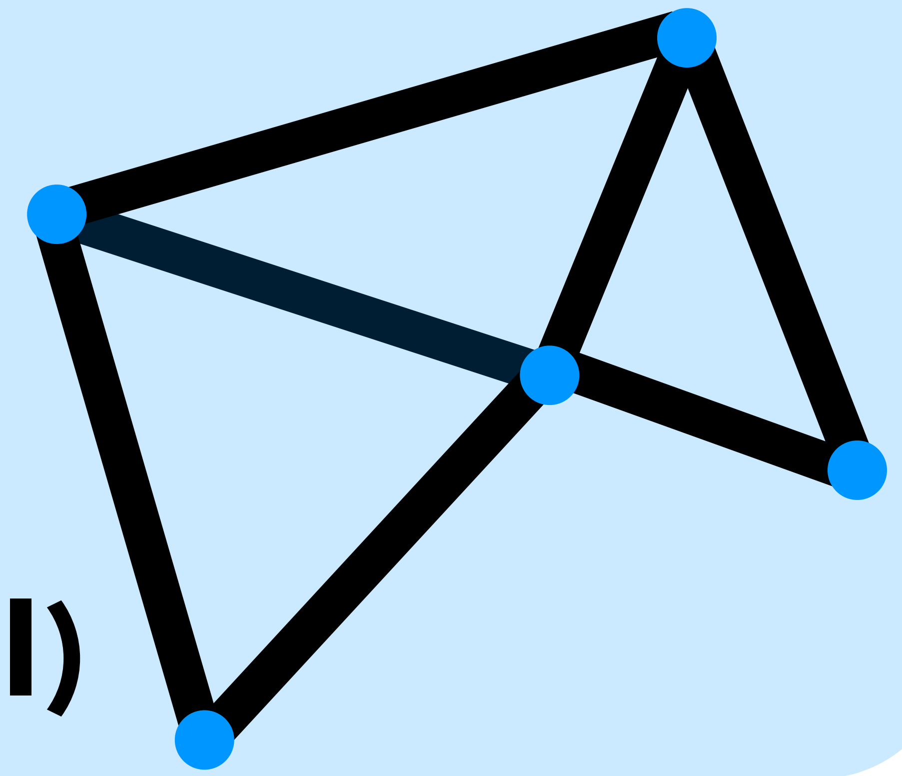


2 ✓ <u>Node sparsification</u>

graph $H = (V' \subseteq V, E')$

$d_H \approx d_G$ on V'

(Steiner Point Removal)

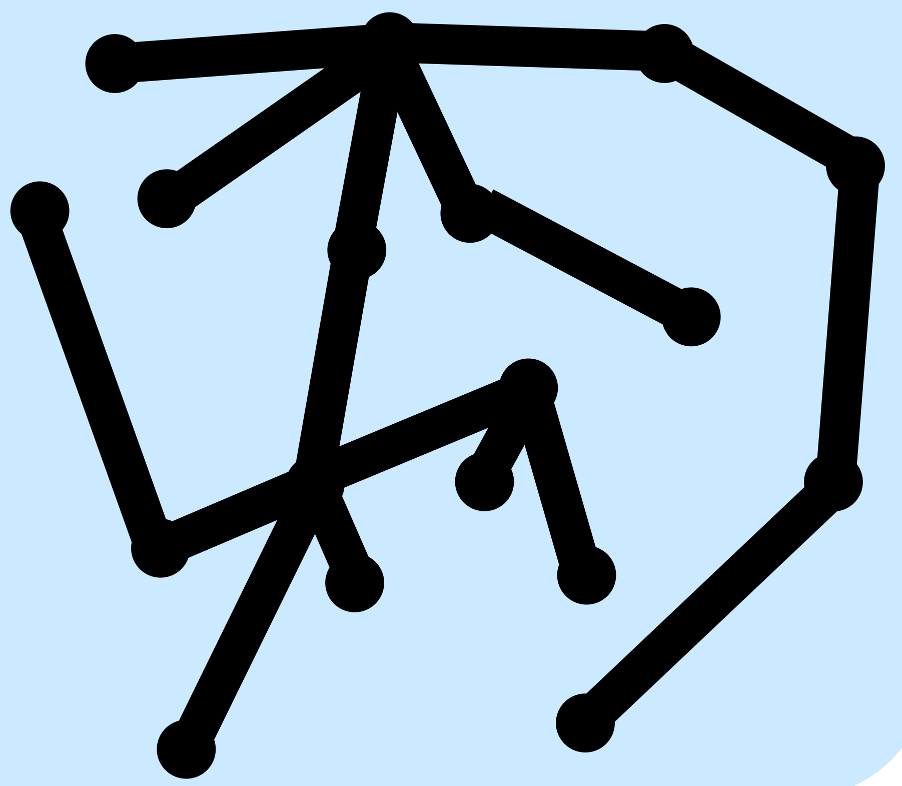


4 <u>Structure sparsification</u>

random tree $T = (V, E')$

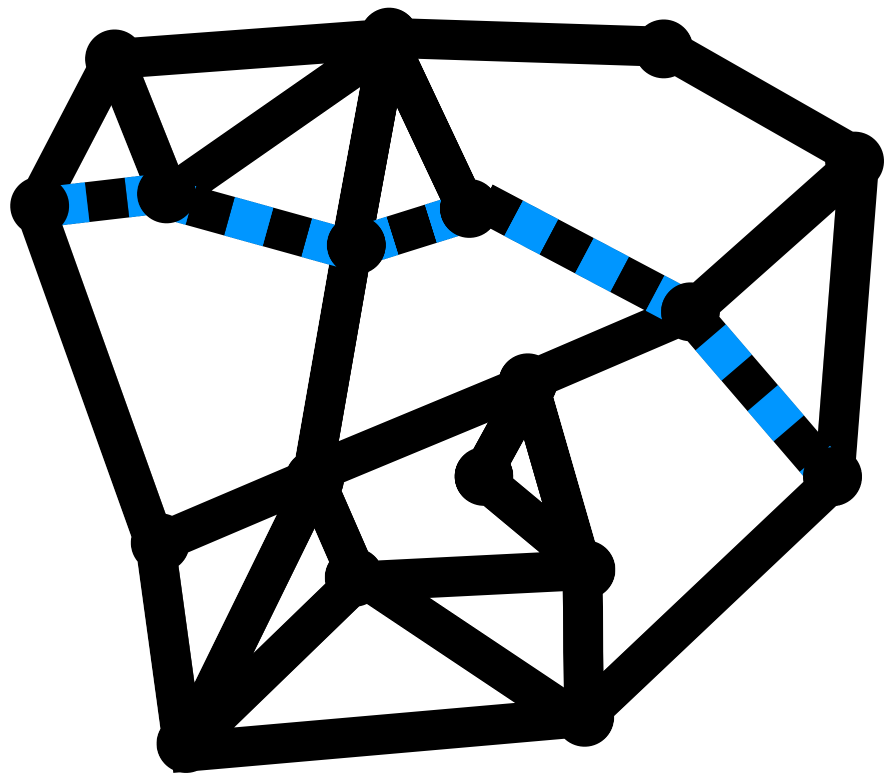
$\mathbb{E}[d_T] \approx d_G$

(Tree Embeddings)



Papers Overview

Distance Sparsification



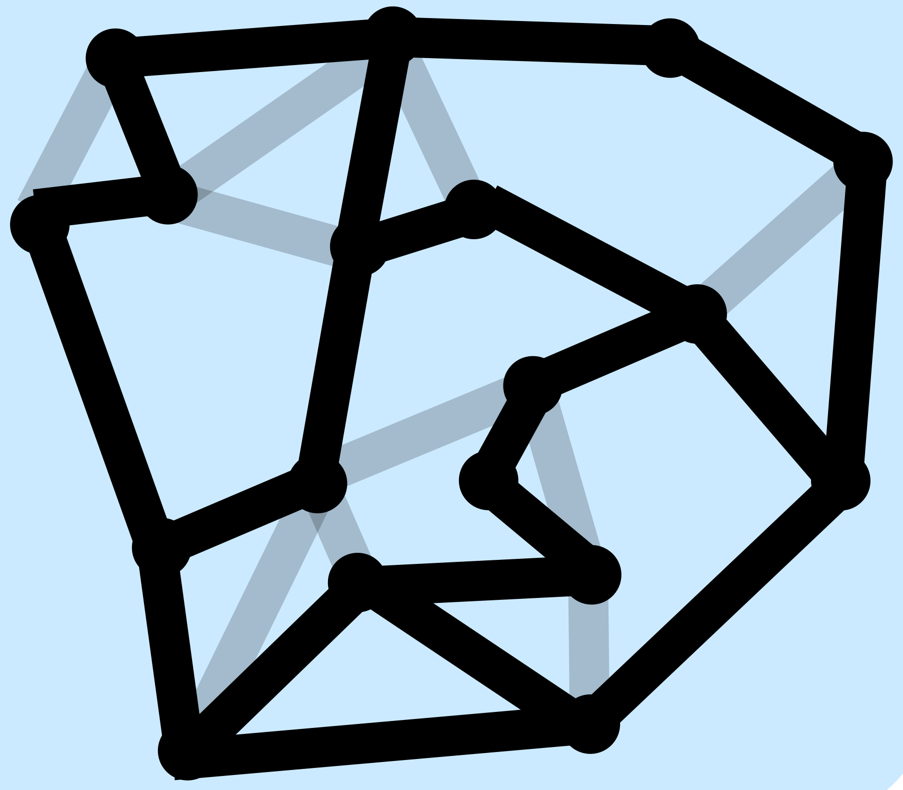
graph $G = (V, E)$

1 ✓ Edge sparsification

graph $H = (V, E' \subseteq E)$

$d_H \approx d_G$

(spanners)

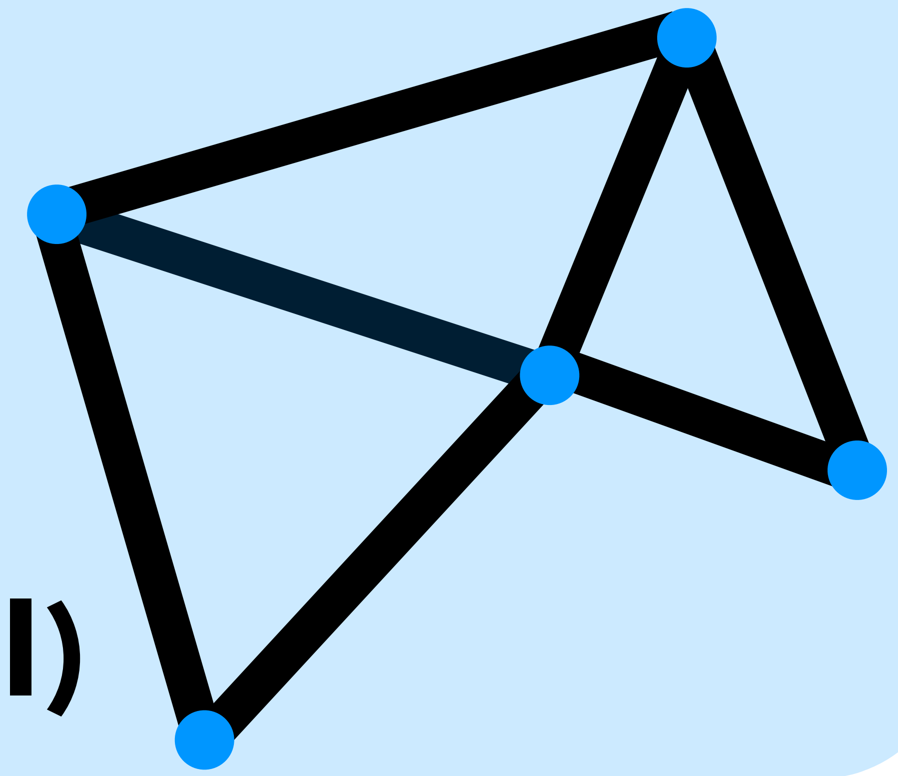


2 ✓ Node sparsification

graph $H = (V' \subseteq V, E')$

$d_H \approx d_G$ on V'

(Steiner Point Removal)



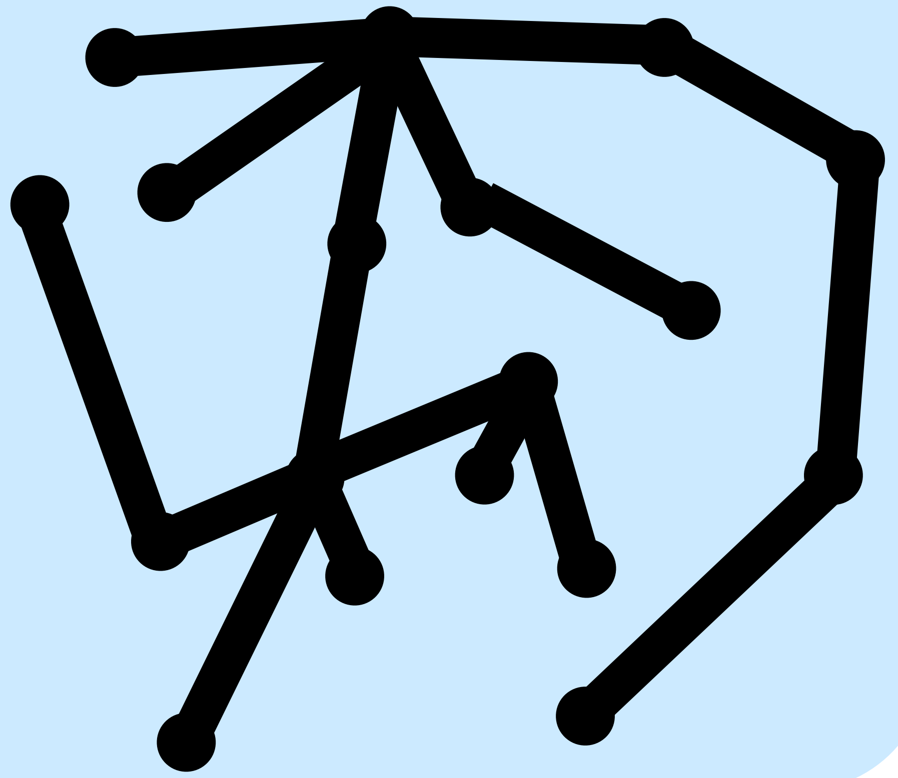
3 ✓

4 Structure sparsification

random tree $T = (V, E')$

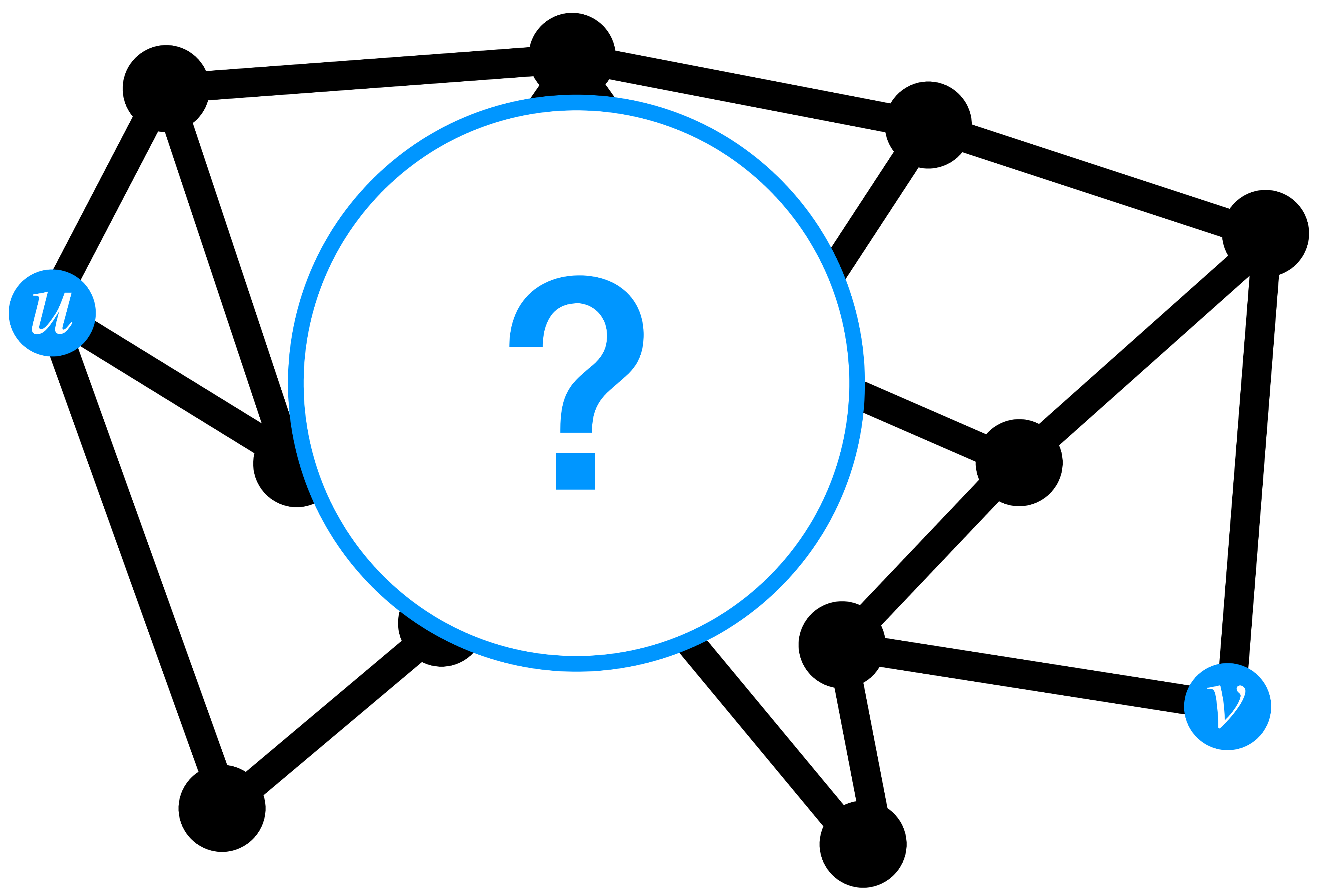
$\mathbb{E}[d_T] \approx d_G$

(Tree Embeddings)

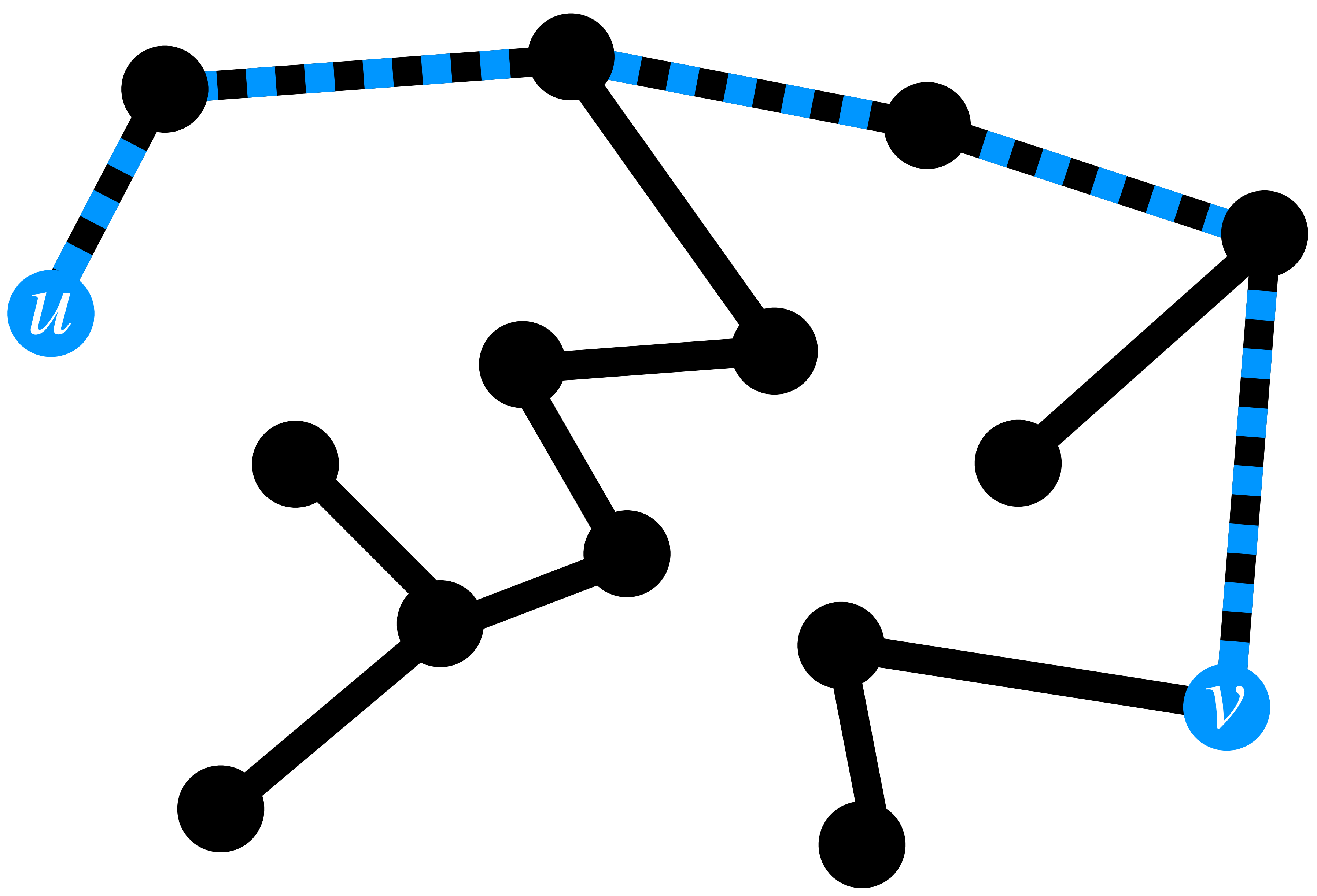


Papers Overview

Paper 4: Tree Embeddings



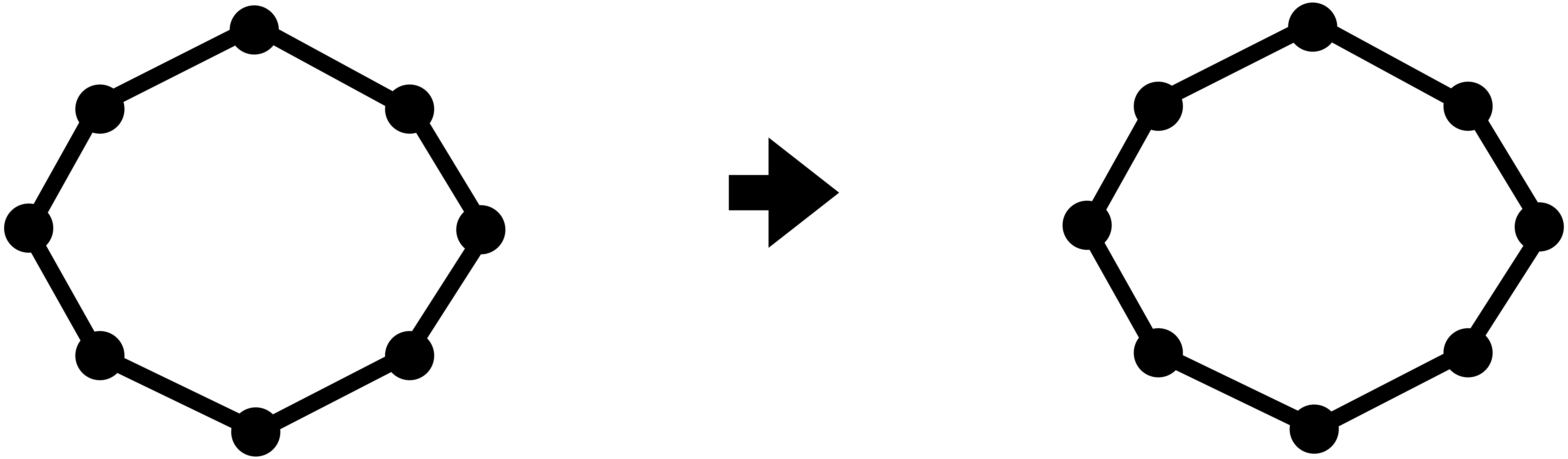
What's the $u \rightarrow v$ shortest path?



What's the $u \rightarrow v$ shortest path?

Papers Overview

Paper 4: Tree Embeddings



graph $G = (V, E)$

tree $T = (V, E', w)$

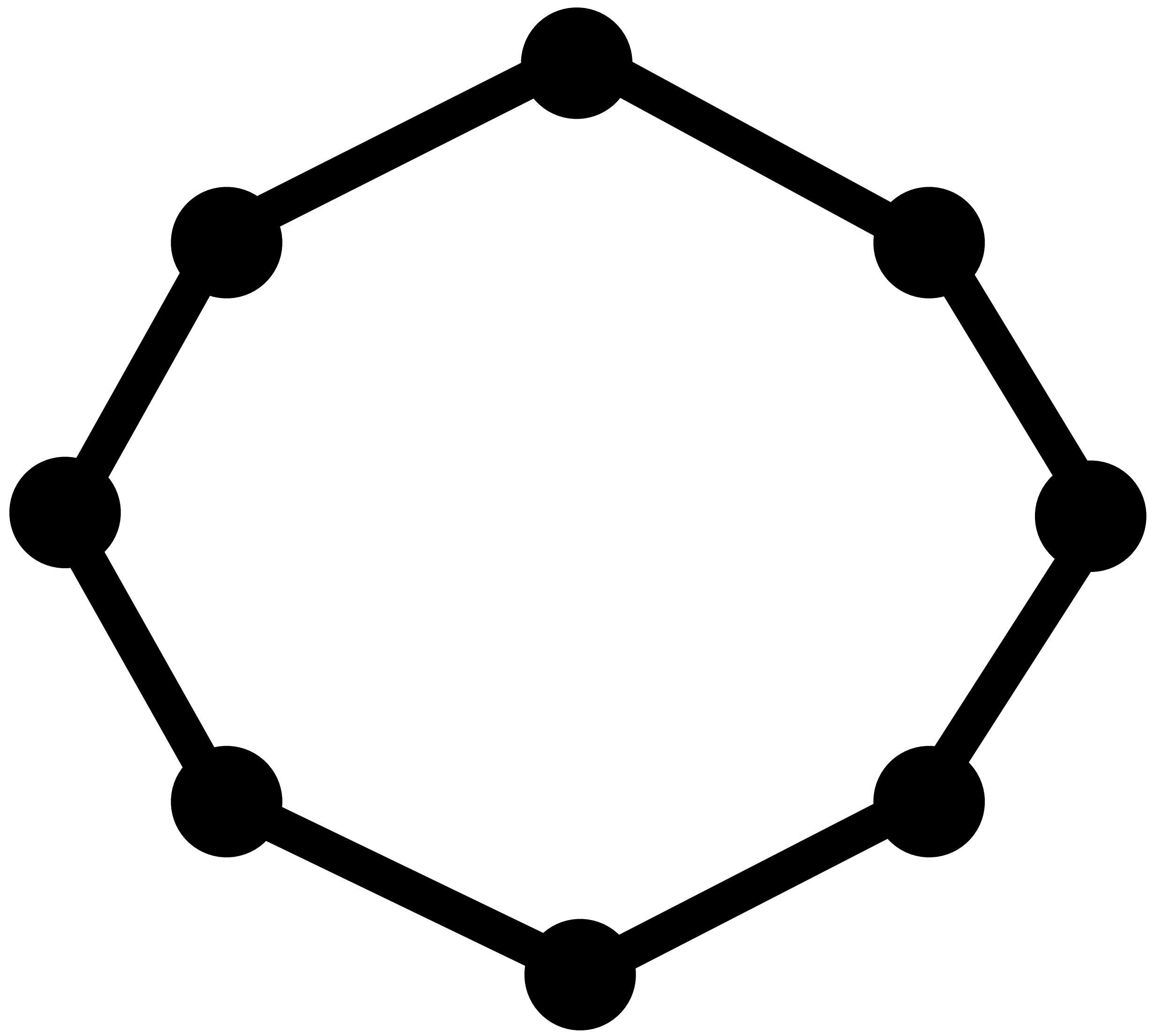
$$d_G(u, v) \leq d_T(u, v) \leq \alpha \cdot d_G(u, v)$$

$\forall u, v \in V$

Goal: approximate arbitrary graph distances by a tree

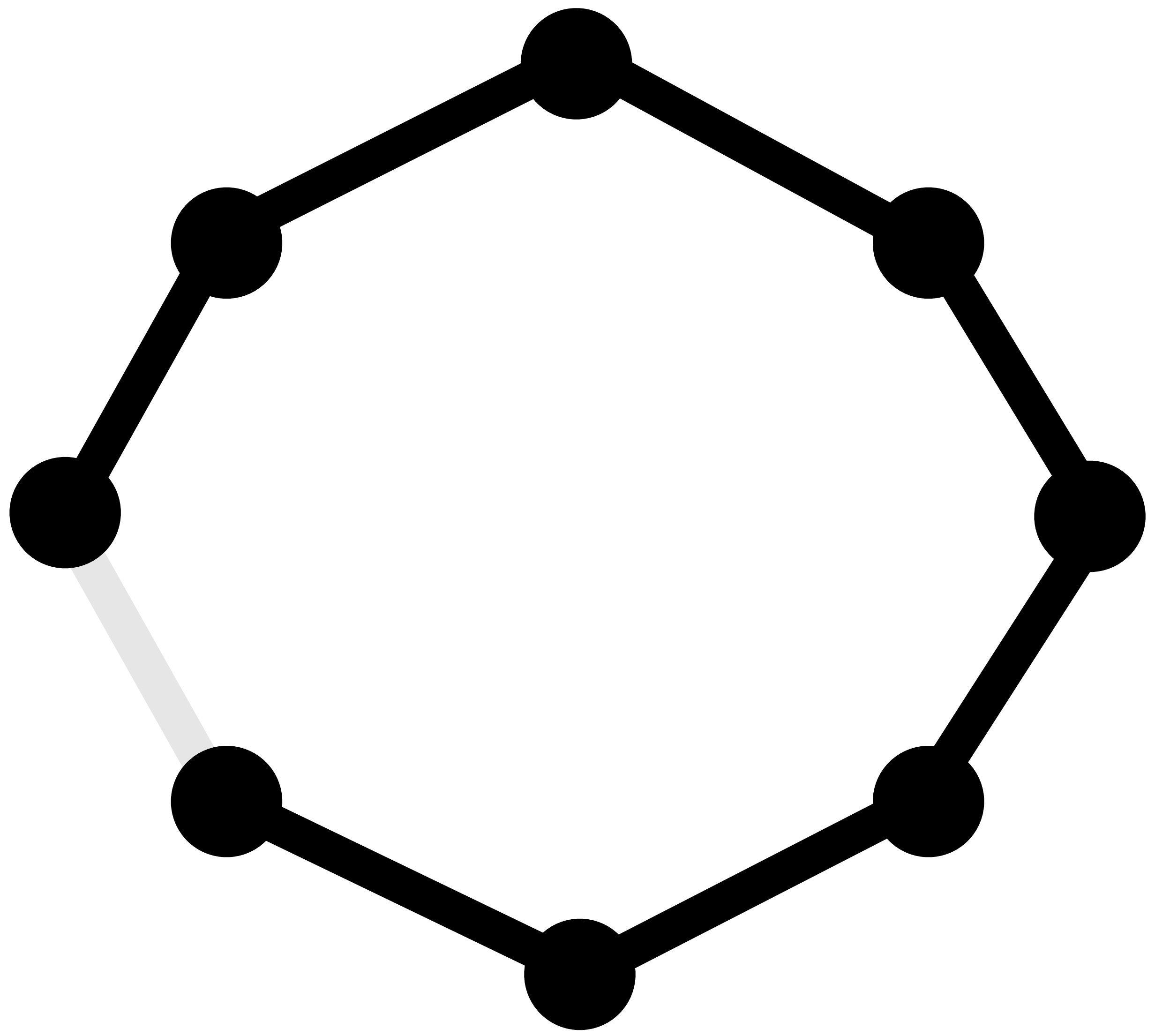
Papers Overview

Paper 4: Tree Embeddings



graph $G = (V, E)$

No hope for a single (spanning) tree!



tree $T = (V, E', w)$

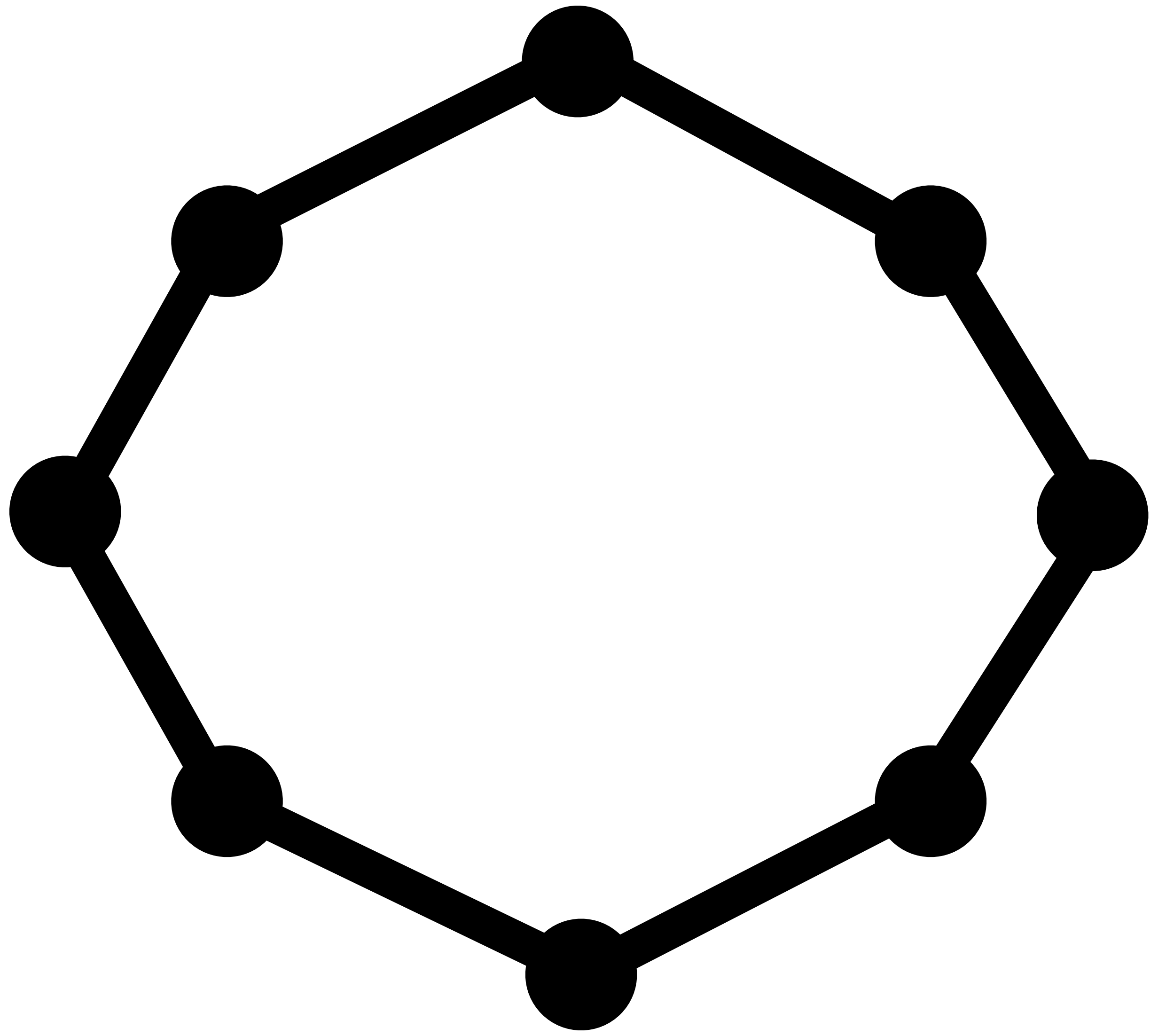
$$d_G(u,v) \le d_T(u,v) \le \alpha \cdot d_G(u,v)$$

$\forall u, v \in V$

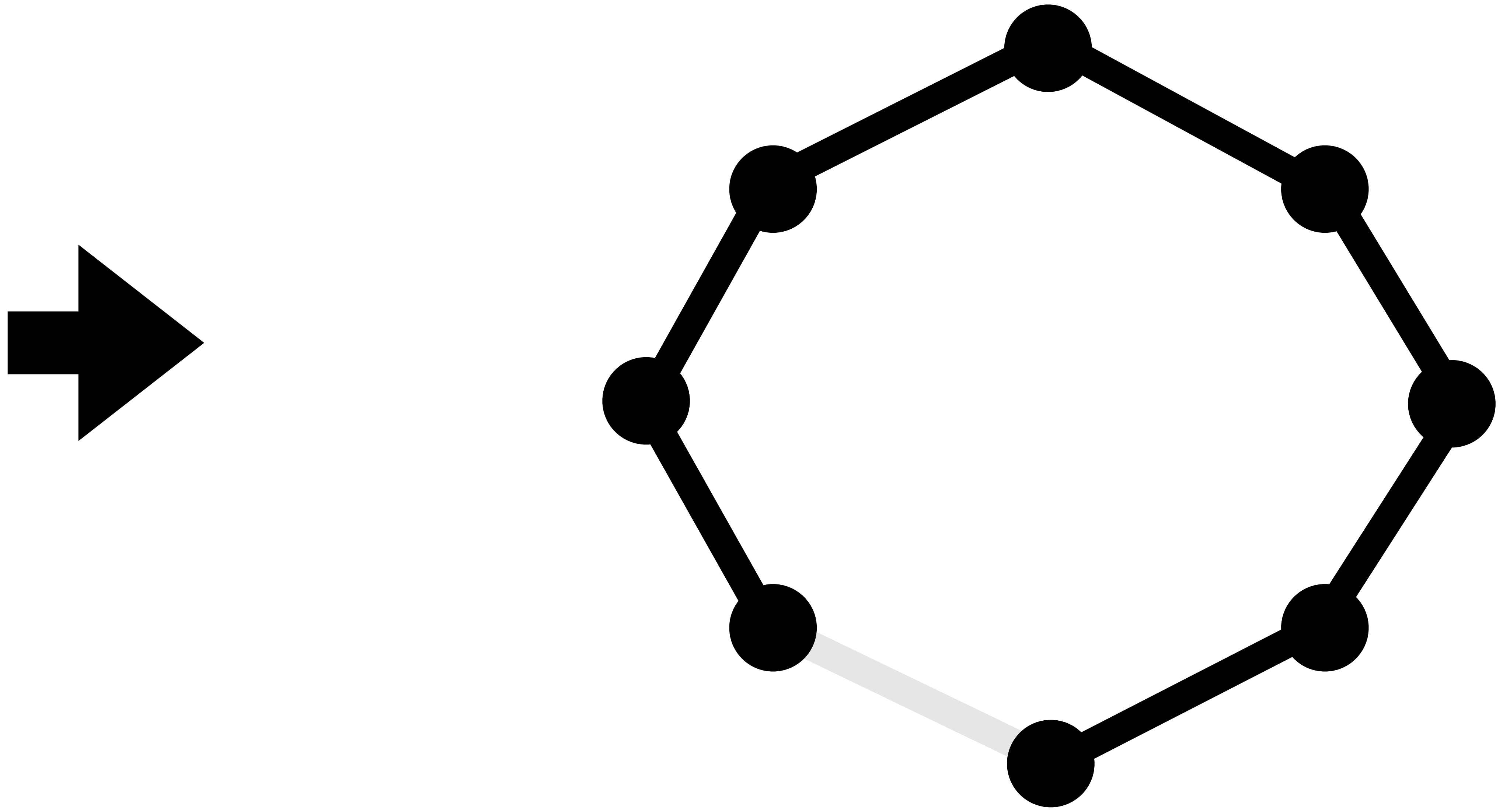
Goal: approximate arbitrary graph distances by a tree

Papers Overview

Paper 4: Tree Embeddings



graph $G = (V, E)$



tree $T = (V, E', w)$

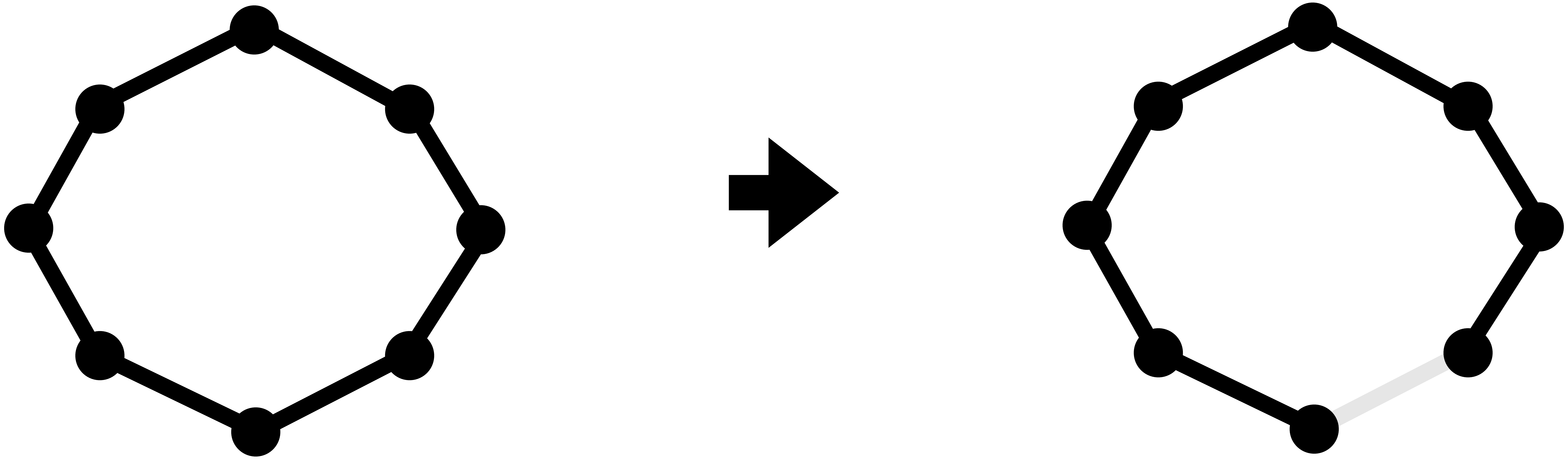
$$d_G(u, v) \leq d_T(u, v) \leq \alpha \cdot d_G(u, v)$$

$$\forall u, v \in V$$

Goal: approximate arbitrary graph distances by a tree

Papers Overview

Paper 4: Tree Embeddings



graph $G = (V, E)$

tree $T = (V, E', w)$

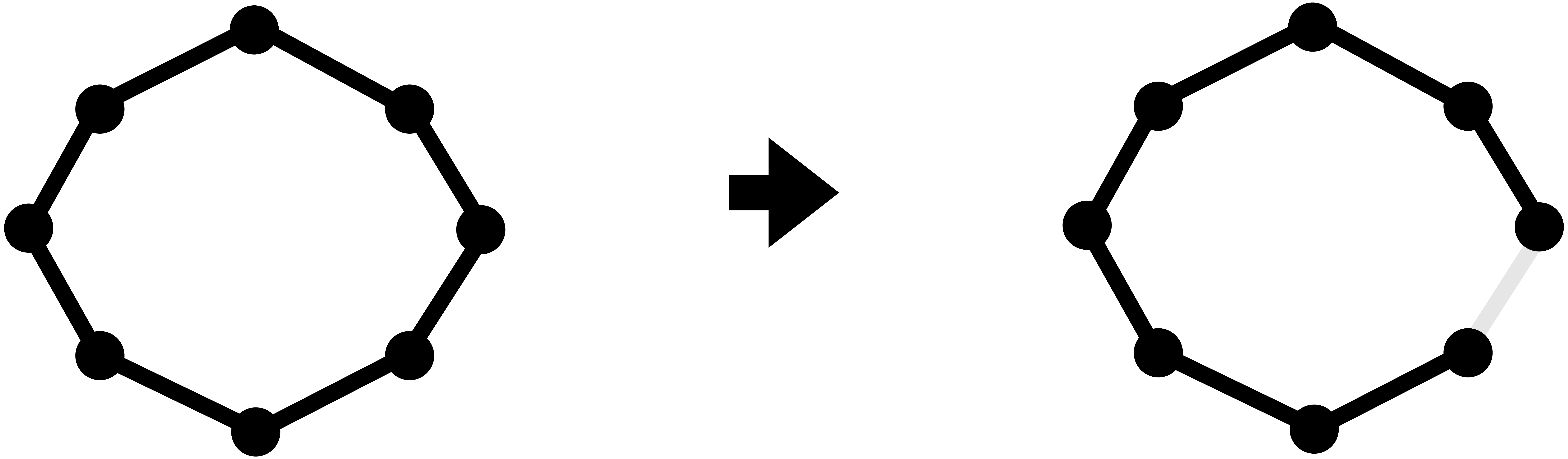
$$d_G(u, v) \leq d_T(u, v) \leq \alpha \cdot d_G(u, v)$$

$$\forall u, v \in V$$

Goal: approximate arbitrary graph distances by a tree

Papers Overview

Paper 4: Tree Embeddings



graph $G = (V, E)$

tree $T = (V, E', w)$

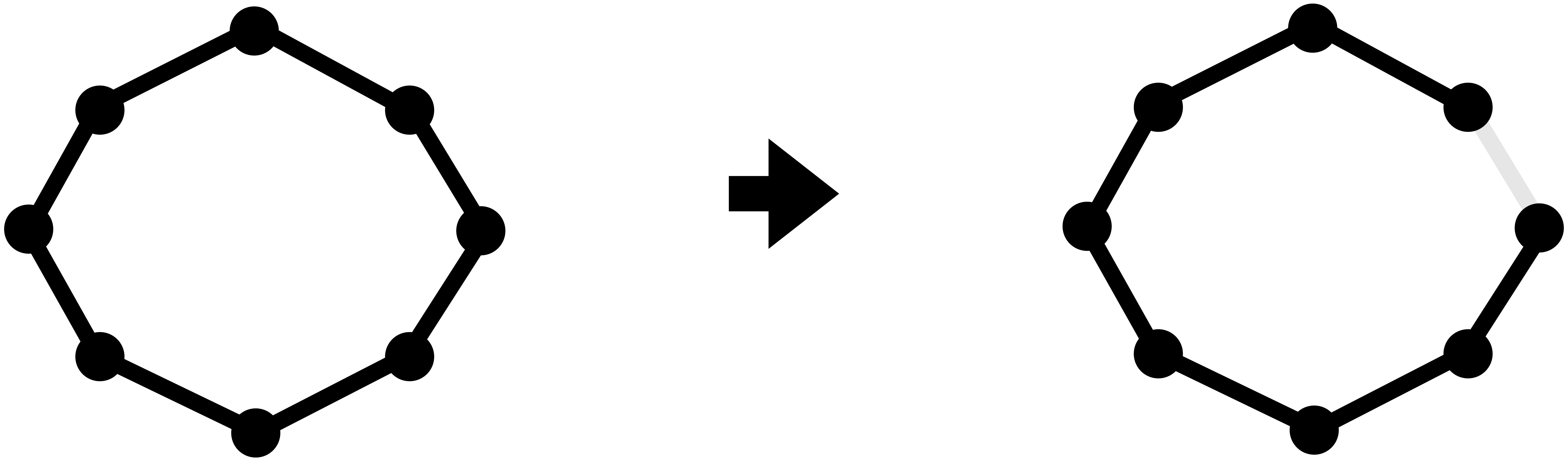
$$d_G(u, v) \leq d_T(u, v) \leq \alpha \cdot d_G(u, v)$$

$\forall u, v \in V$

Goal: approximate arbitrary graph distances by a tree

Papers Overview

Paper 4: Tree Embeddings



graph $G = (V, E)$

tree $T = (V, E', w)$

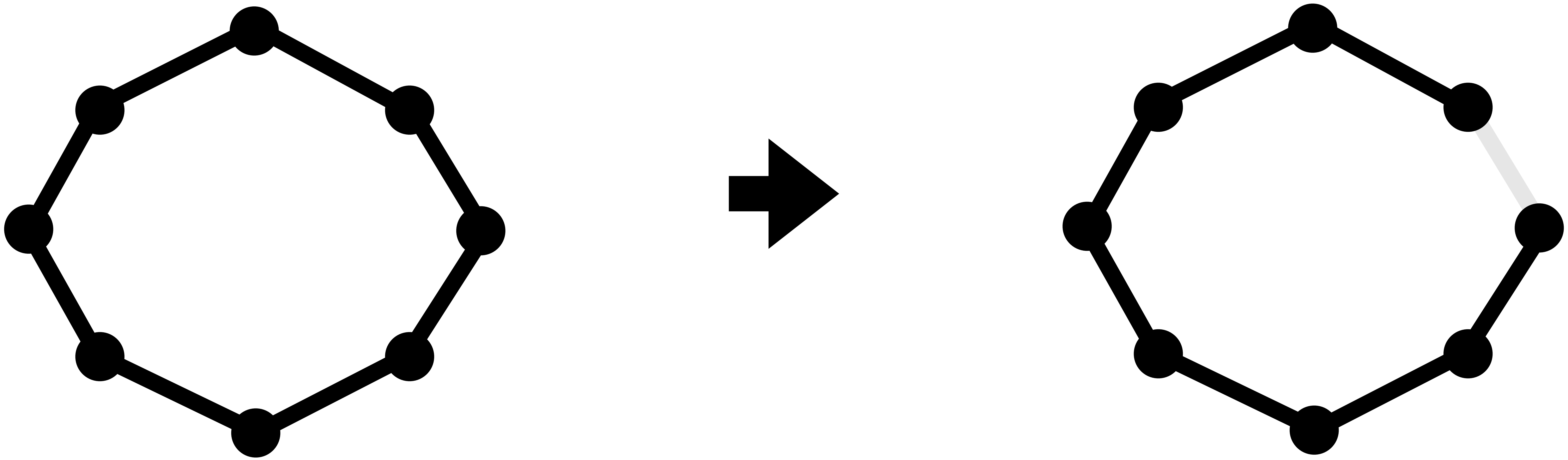
$$d_G(u,v) \leq d_T(u,v) \leq \alpha \cdot d_G(u,v)$$

$\forall u, v \in V$

Goal: approximate arbitrary graph distances by a tree

Papers Overview

Paper 4: Tree Embeddings



graph $G = (V, E)$

distribution on tree T

$$d_G(u, v) \leq \mathbb{E}_T[d_T(u, v)] \leq \alpha \cdot d_G(u, v)$$

$$\forall u, v \in V$$

Goal: approximate arbitrary graph distances by a tree

Papers Overview

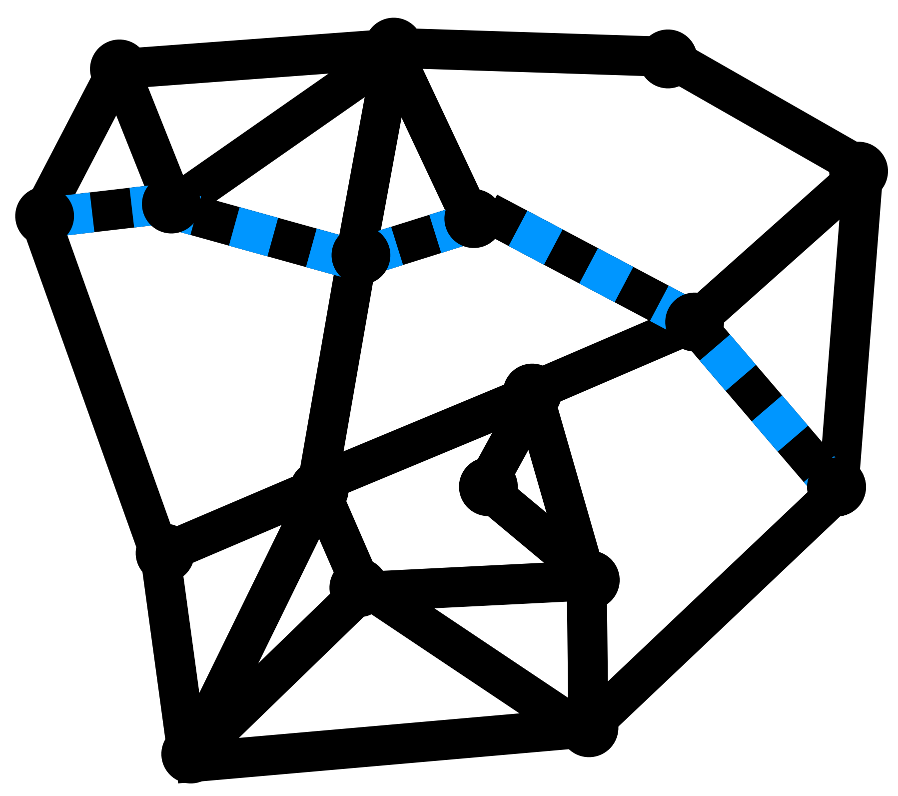
Paper 4: Tree Embeddings

Theorem: Given graph $G = (V, E)$, $\exists$ a distribution $\mathcal{T}$ over trees on V on s.t.

1. $d_G(u, v) \leq d_T(u, v)$ $\quad \forall T \in \mathcal{T}$ and $u, v \in V$ (countless applications)
2. $\mathbb{E}_{T \sim \mathcal{T}}[d_T(u, v)] \leq O(\log n) \cdot d_G(u, v)$ $\quad \forall u, v \in V$

Papers Overview

Distance Sparsification



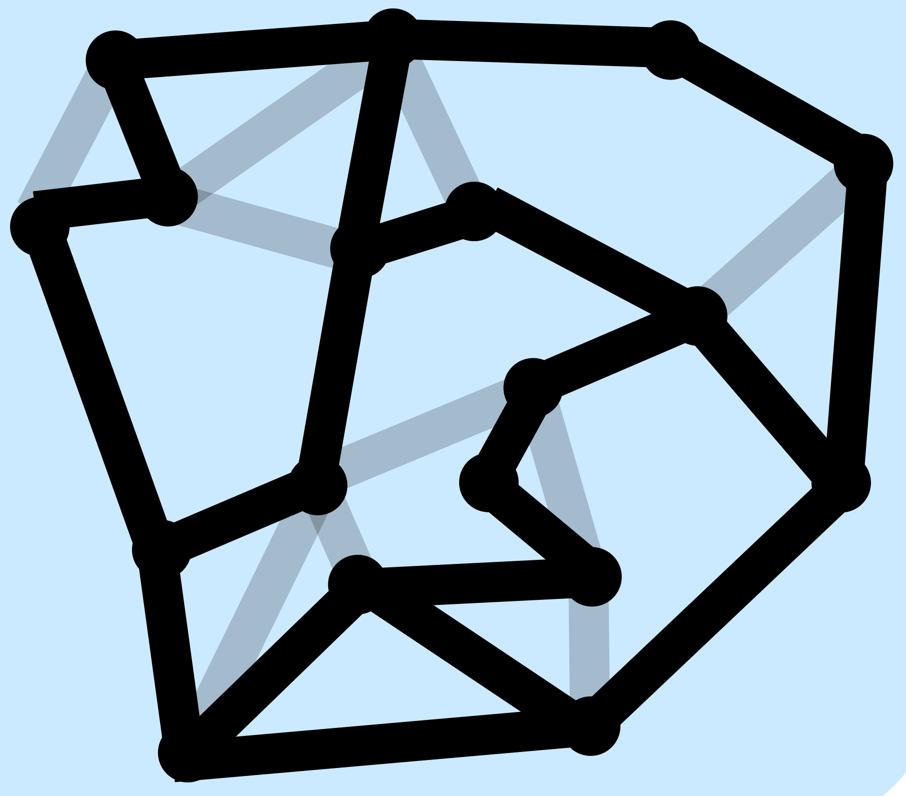
graph $G = (V, E)$

1. Edge sparsification

graph $H = (V, E' \subseteq E)$

$d_H \approx d_G$

(spanners)

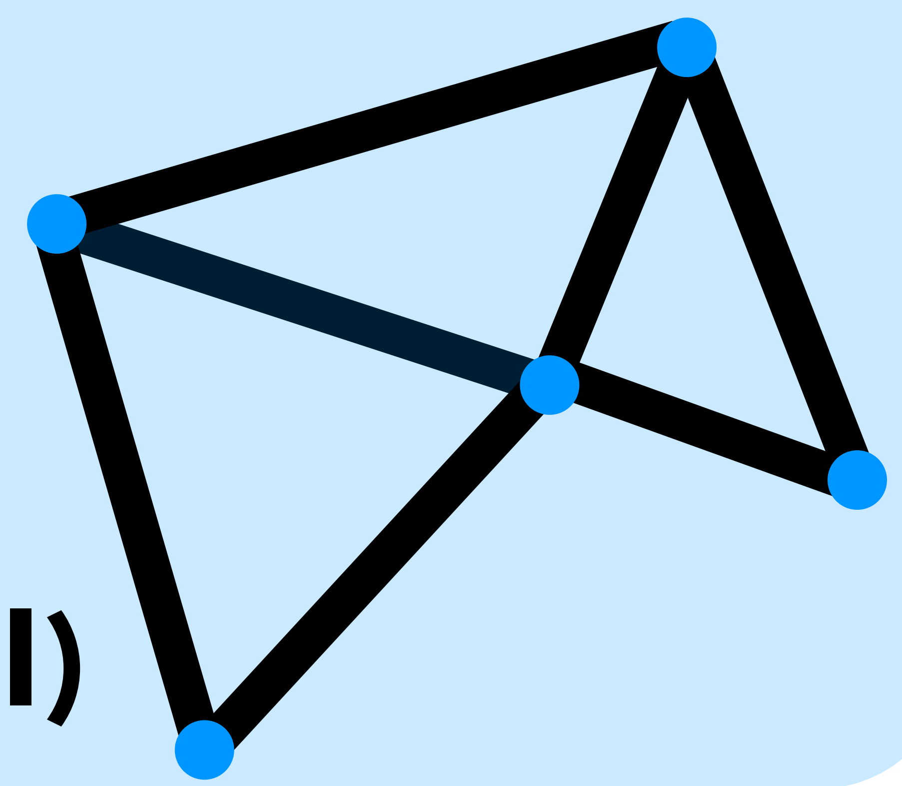


2. Node sparsification

graph $H = (V' \subseteq V, E')$

$d_H \approx d_G$ on V'

(Steiner Point Removal)



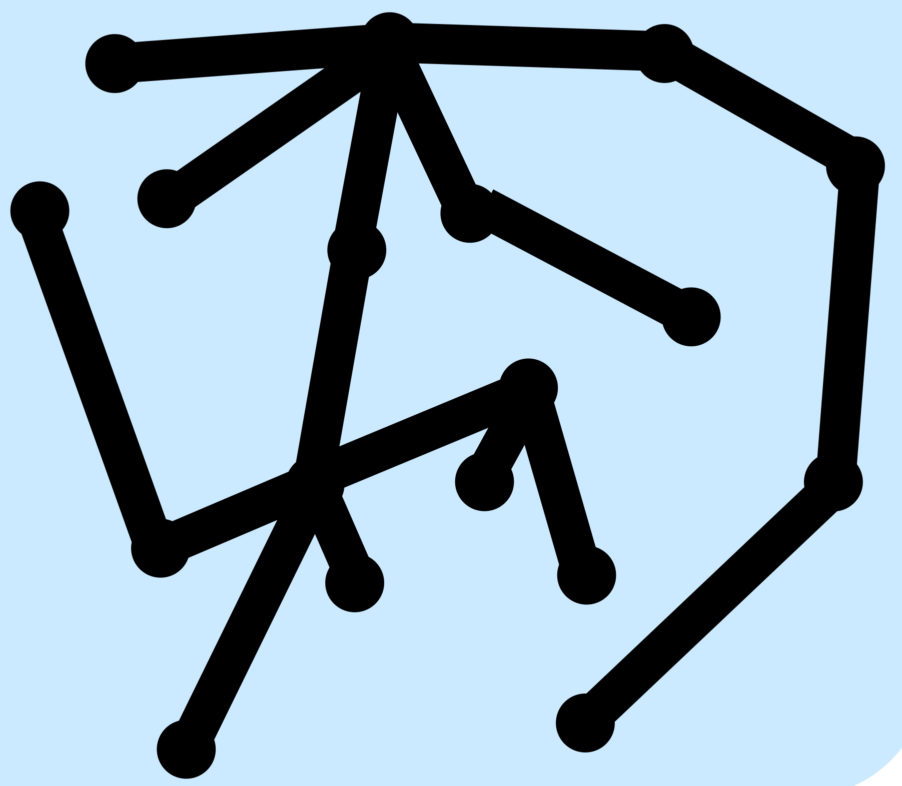
3.

4. Structure sparsification

random tree $T = (V, E')$

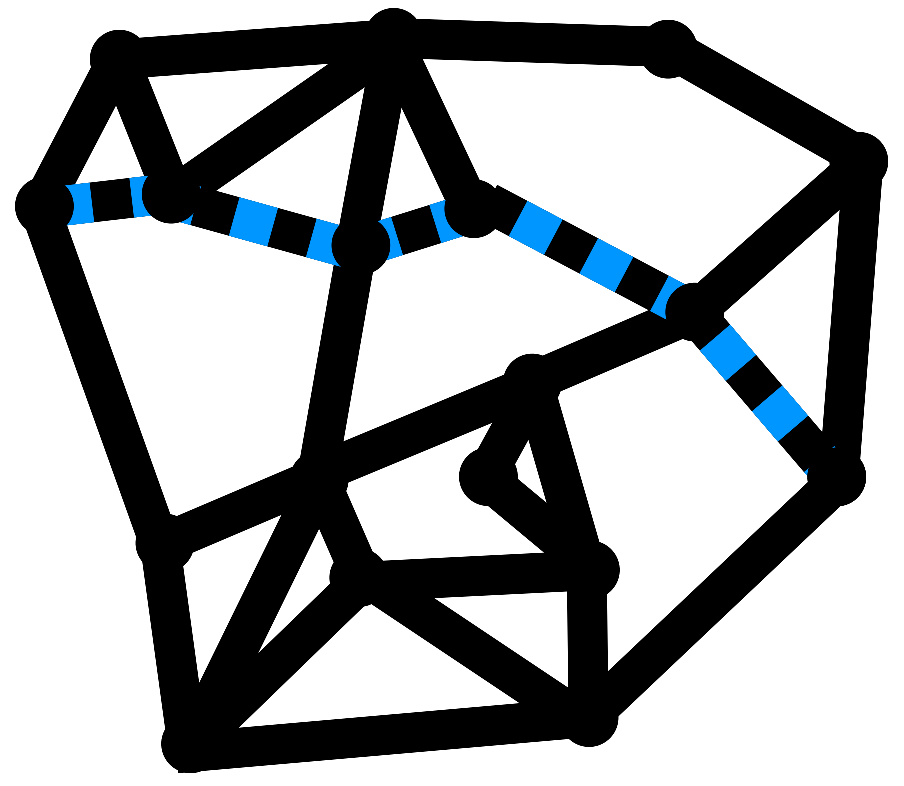
$\mathbb{E}[d_T] \approx d_G$

(Tree Embeddings)



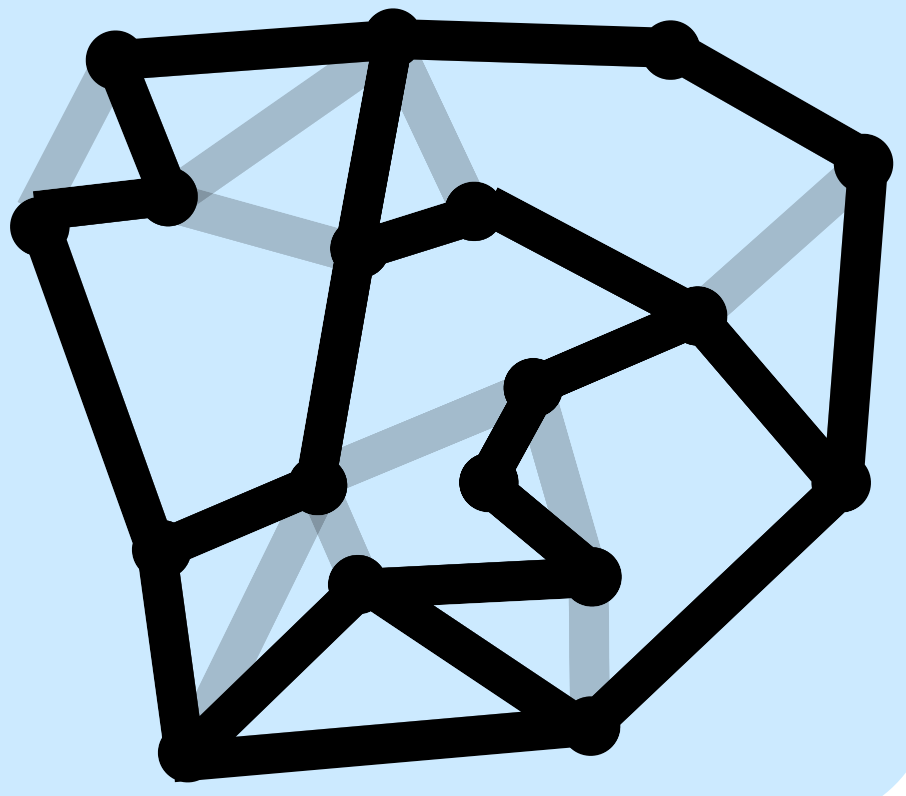
Papers Overview

Distance Sparsification

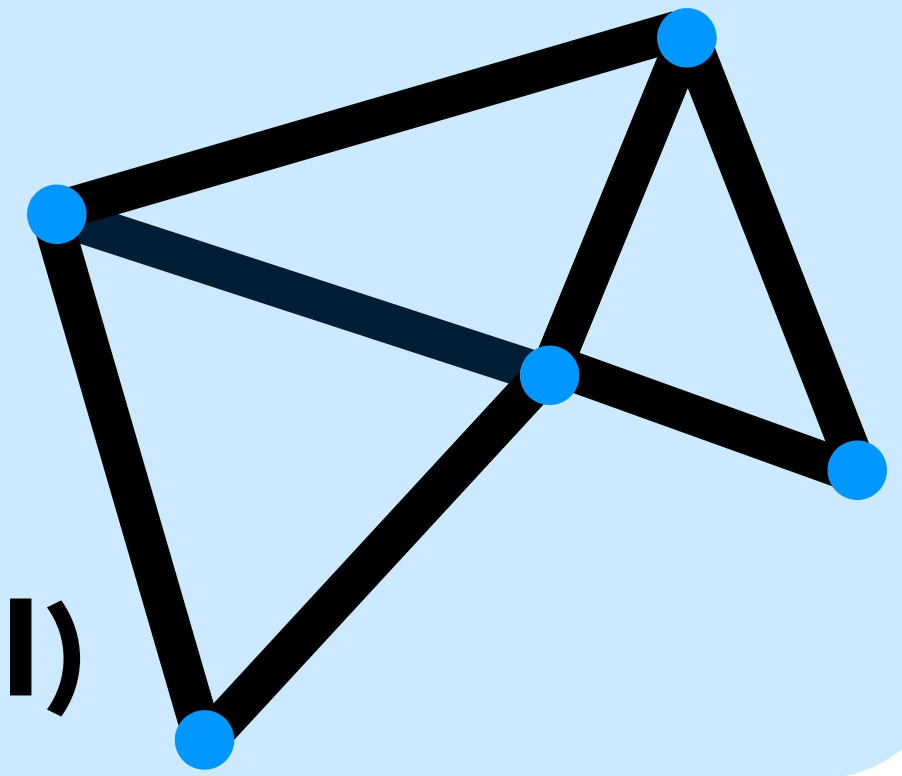


graph $G = (V, E)$

1. Edge sparsification
   graph $H = (V, E' \subseteq E)$
   $d_H \approx d_G$
   (spanners)

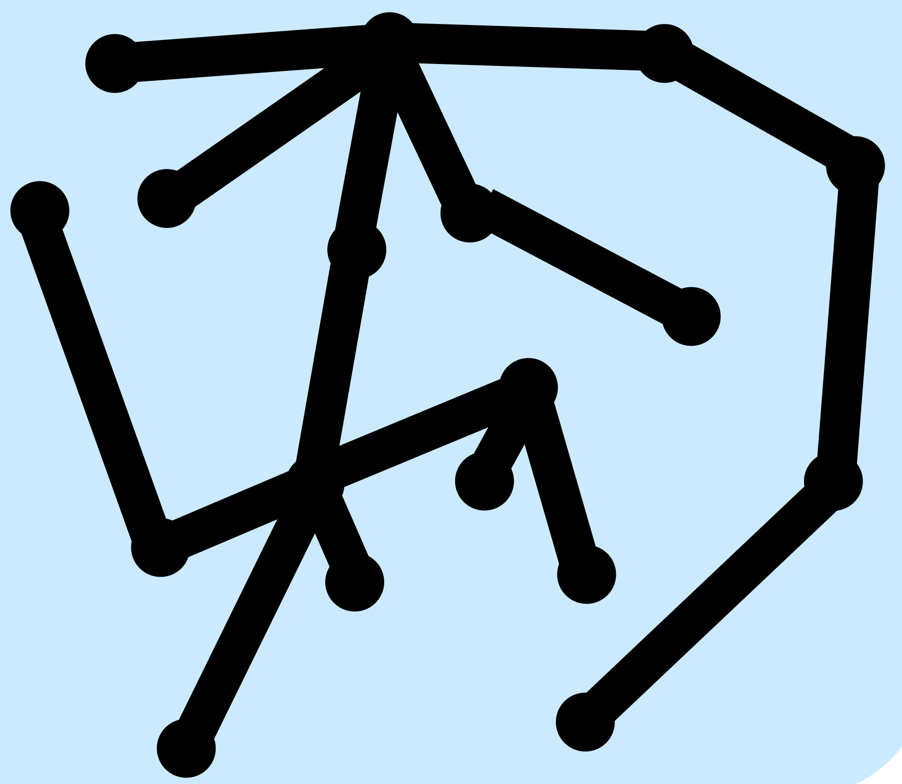


2. Node sparsification
   graph $H = (V' \subseteq V, E')$
   $d_H \approx d_G$ on V'
   (Steiner Point Removal)



3.

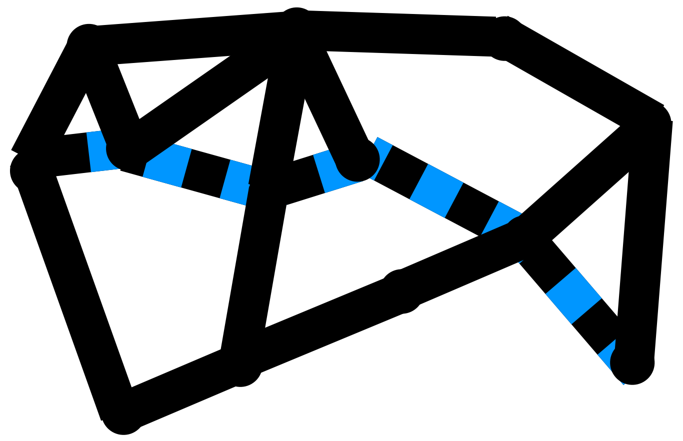
4. Structure sparsification
   random tree $T = (V, E')$
   $\mathbb{E}[d_T] \approx d_G$
   (Tree Embeddings)



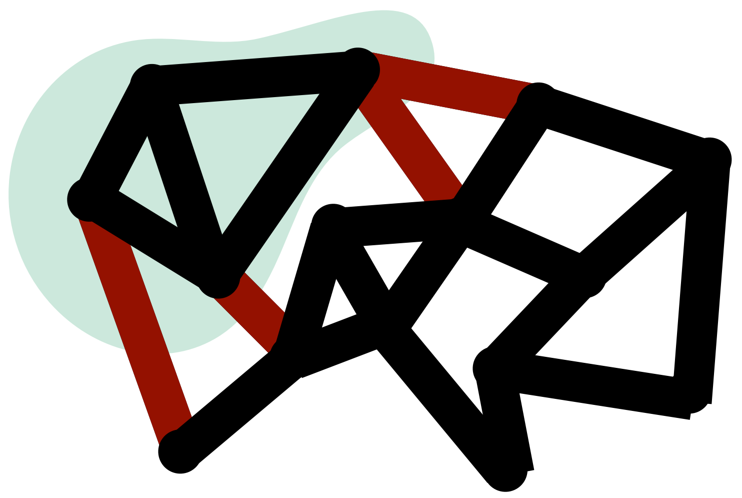
Papers Overview

Sparsification of Five Graph-Theoretic Objects

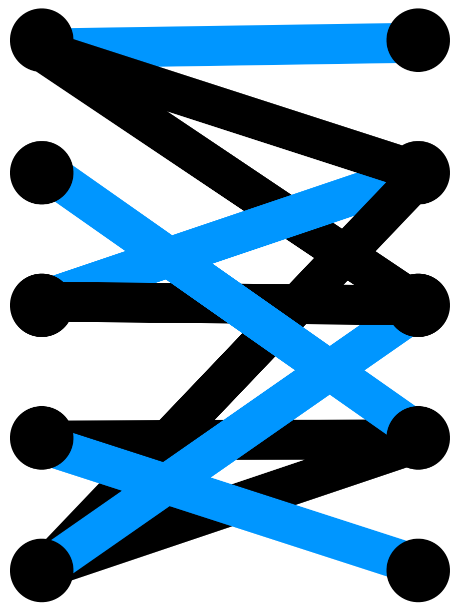
Distances



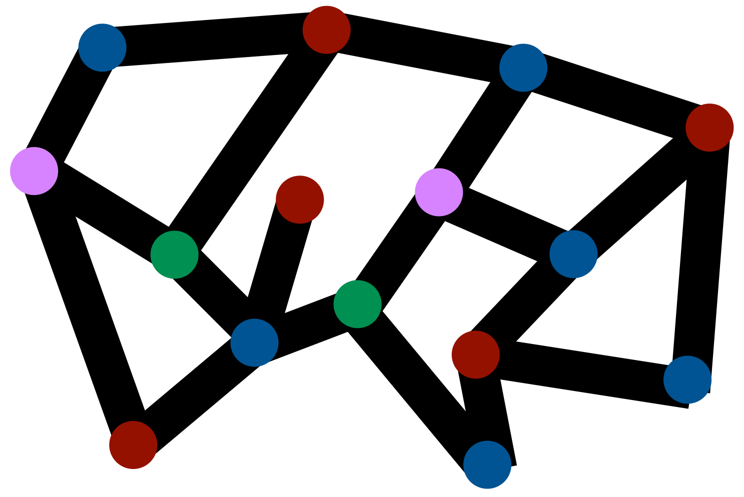
Cuts/Flows



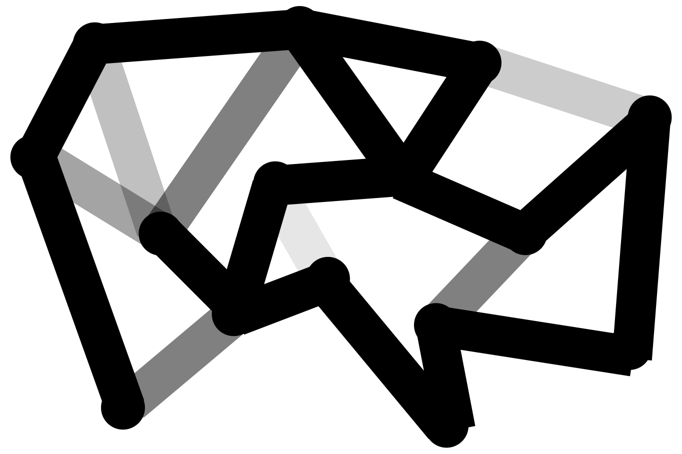
Matchings



Colorings



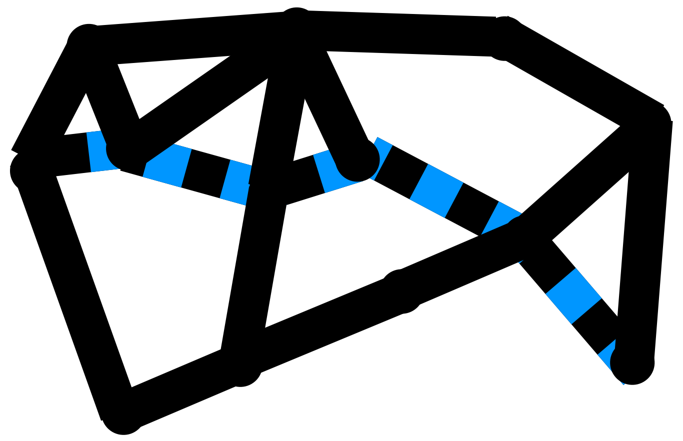
Fractional Opts



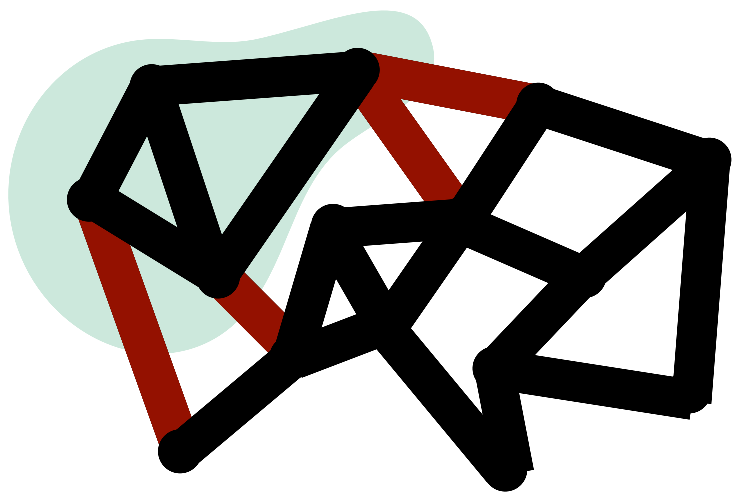
Papers Overview

Sparsification of Five Graph-Theoretic Objects

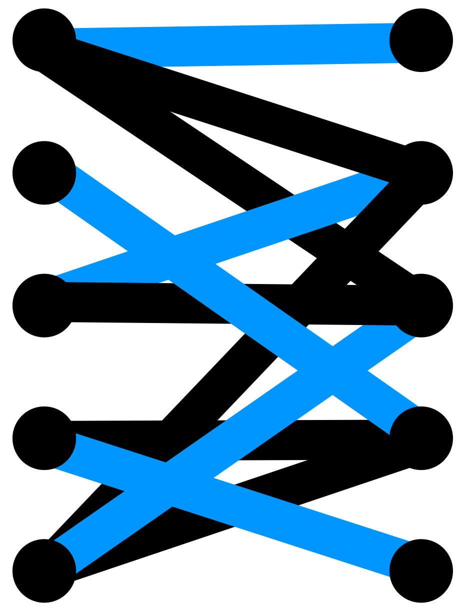
Distances ✓



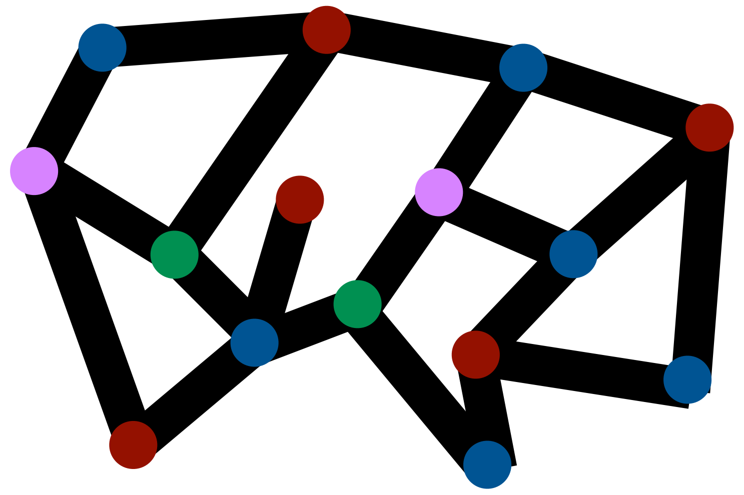
Cuts/Flows



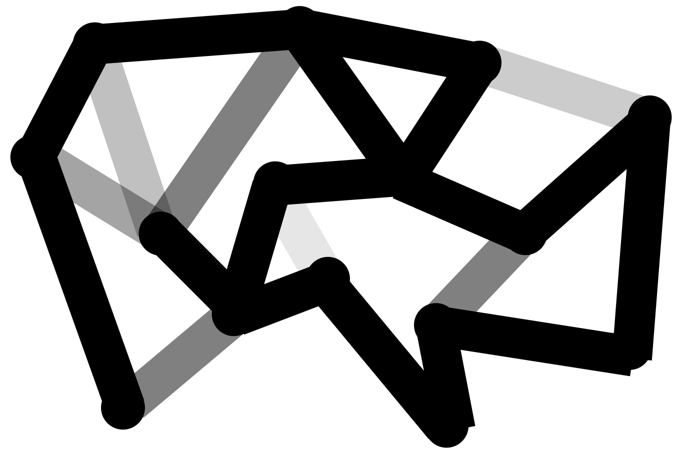
Matchings



Colorings

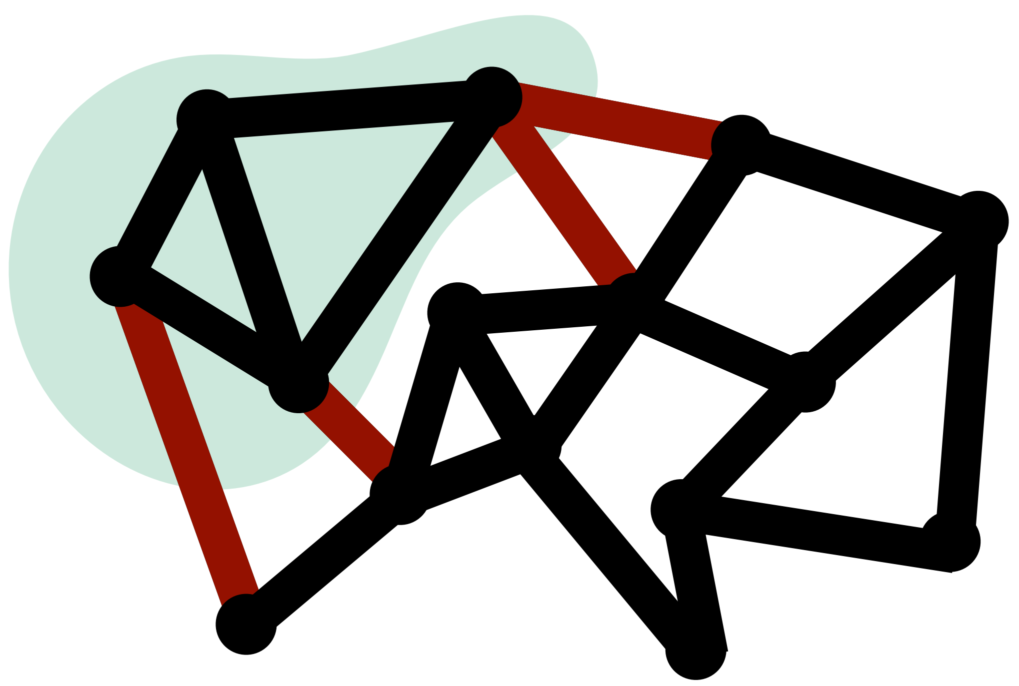


Fractional Opts



Papers Overview

Flow / Cut Sparsification



graph $G = (V, E)$

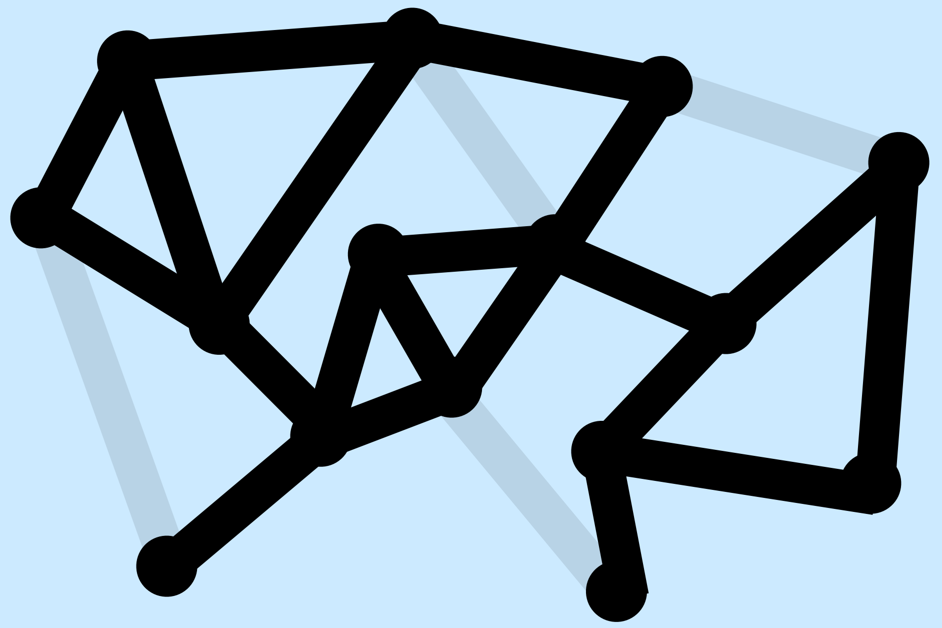
(4 ✓) → 5, 6, 7 → 8

5 **Edge sparsification**

graph $H = (V, E' \subseteq E)$

H cuts $\approx G$ cuts

(Random Sampling)

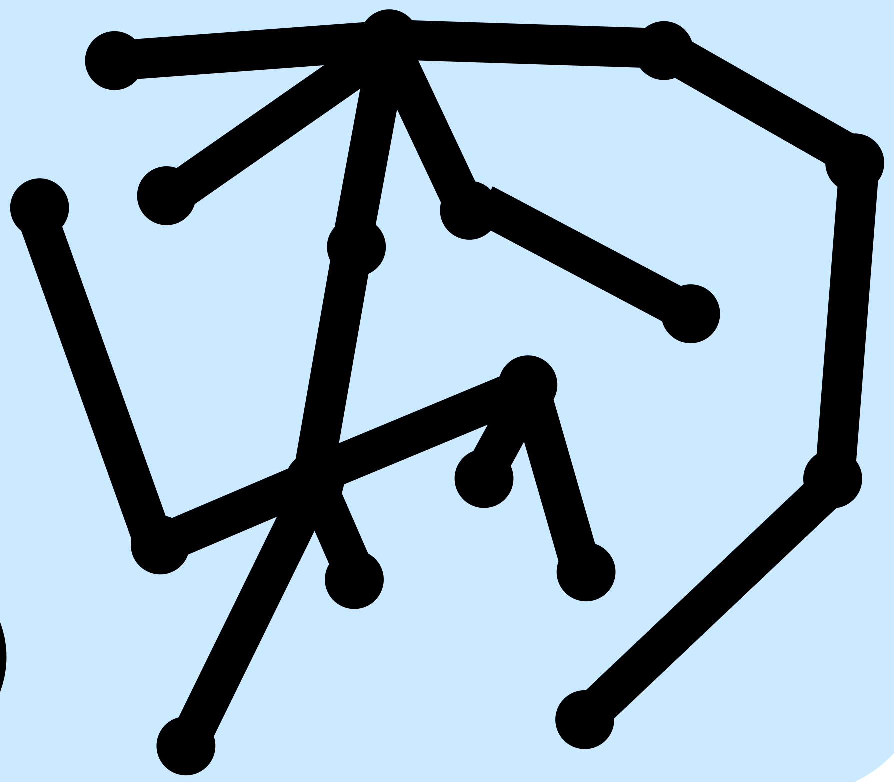


6 **Structure sparsification**

tree $T = (V, E')$

T flows $\approx G$ flows

(Tree Flow Sparsifiers)

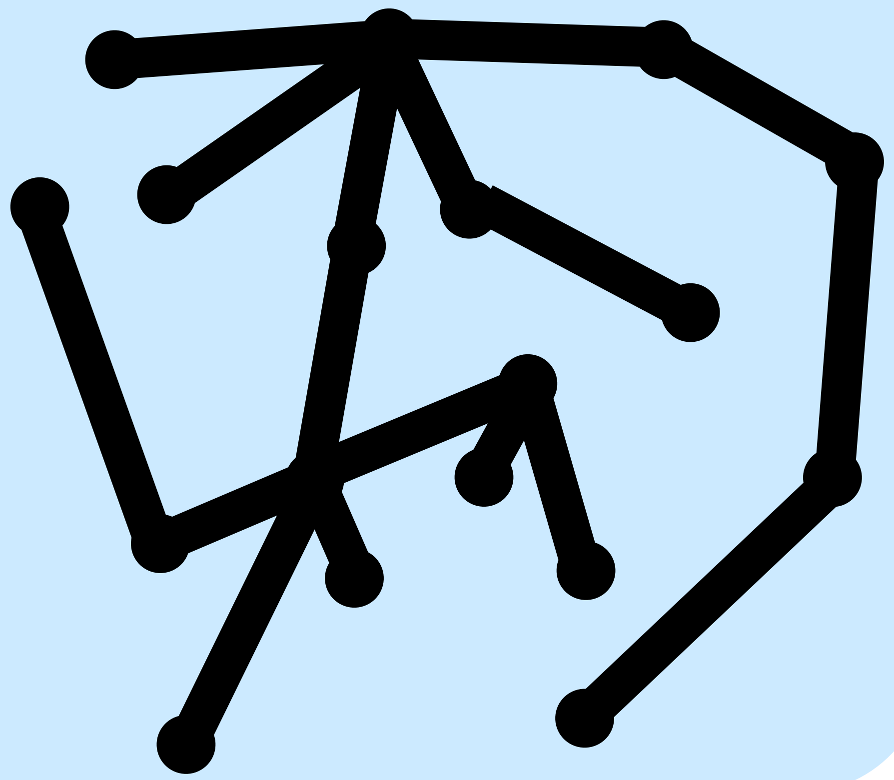


8 **Dynamic sparsification**

tree $T = (V, E')$

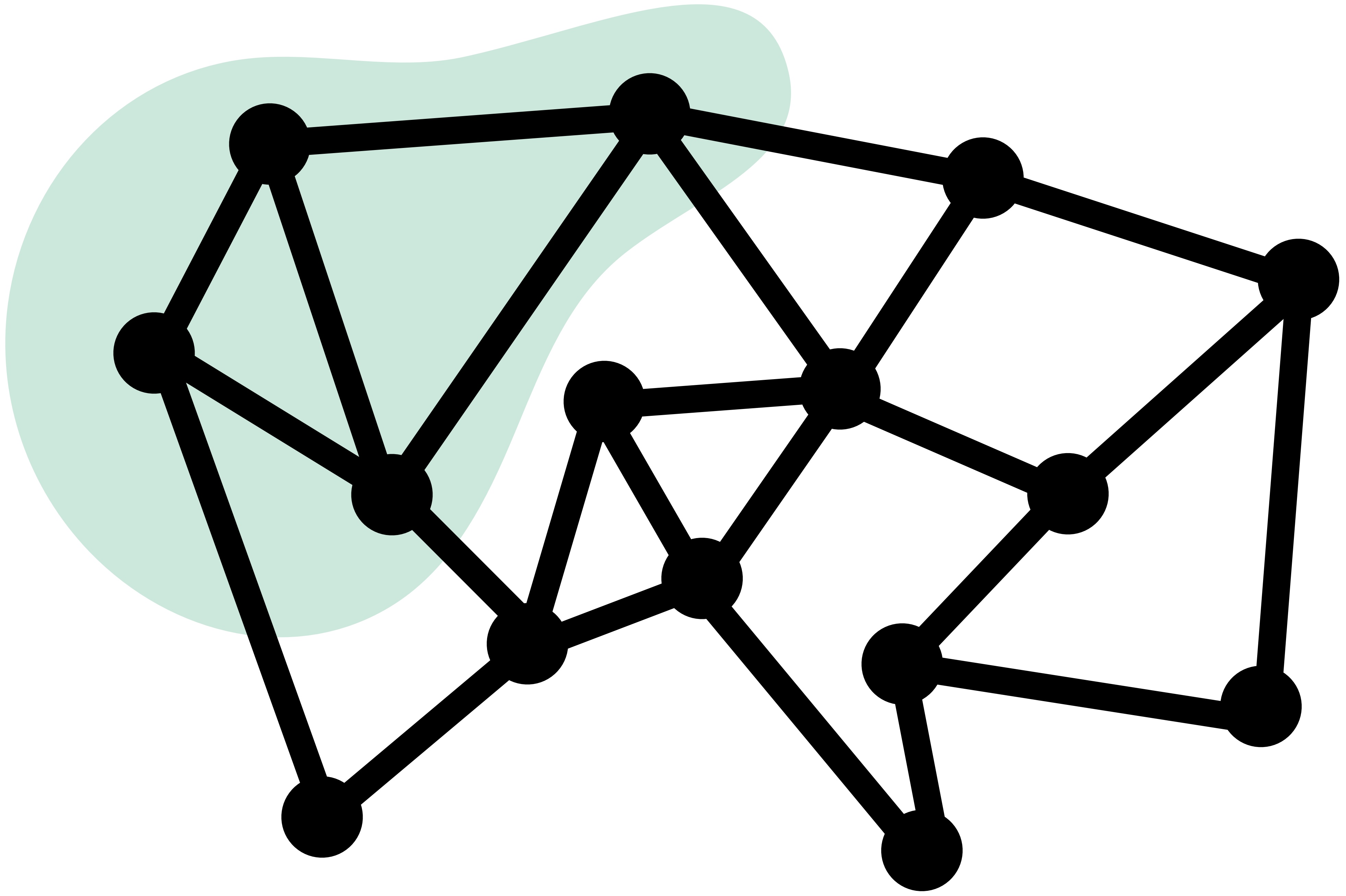
T flows $\approx G$ flows

(Dynamic Tree Flow Sparsifiers)



Papers Overview

Background: Cuts

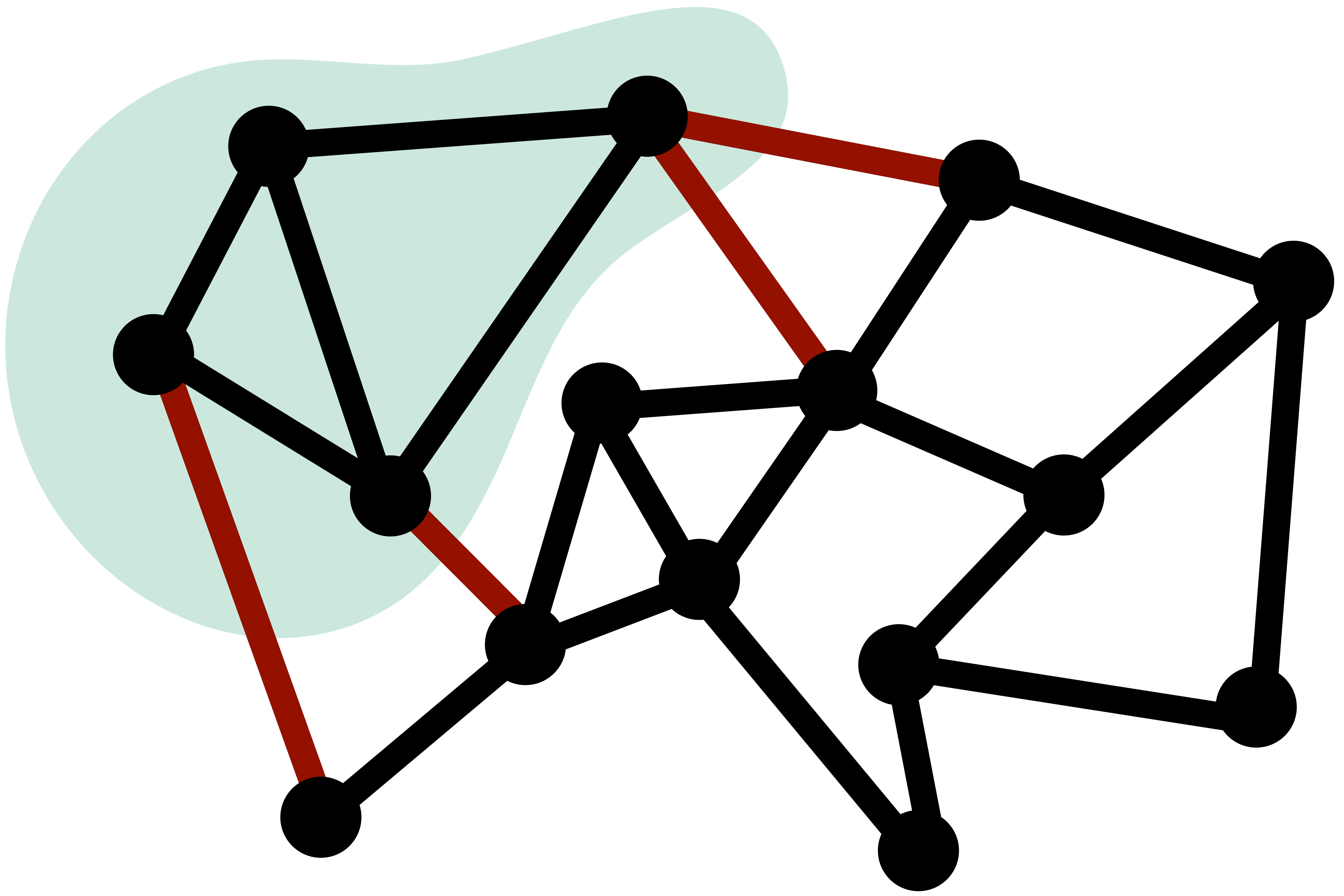


graph $G = (V, E)$

Definition (Cut)**:** any $S \subseteq V$

Papers Overview

Background: Cuts

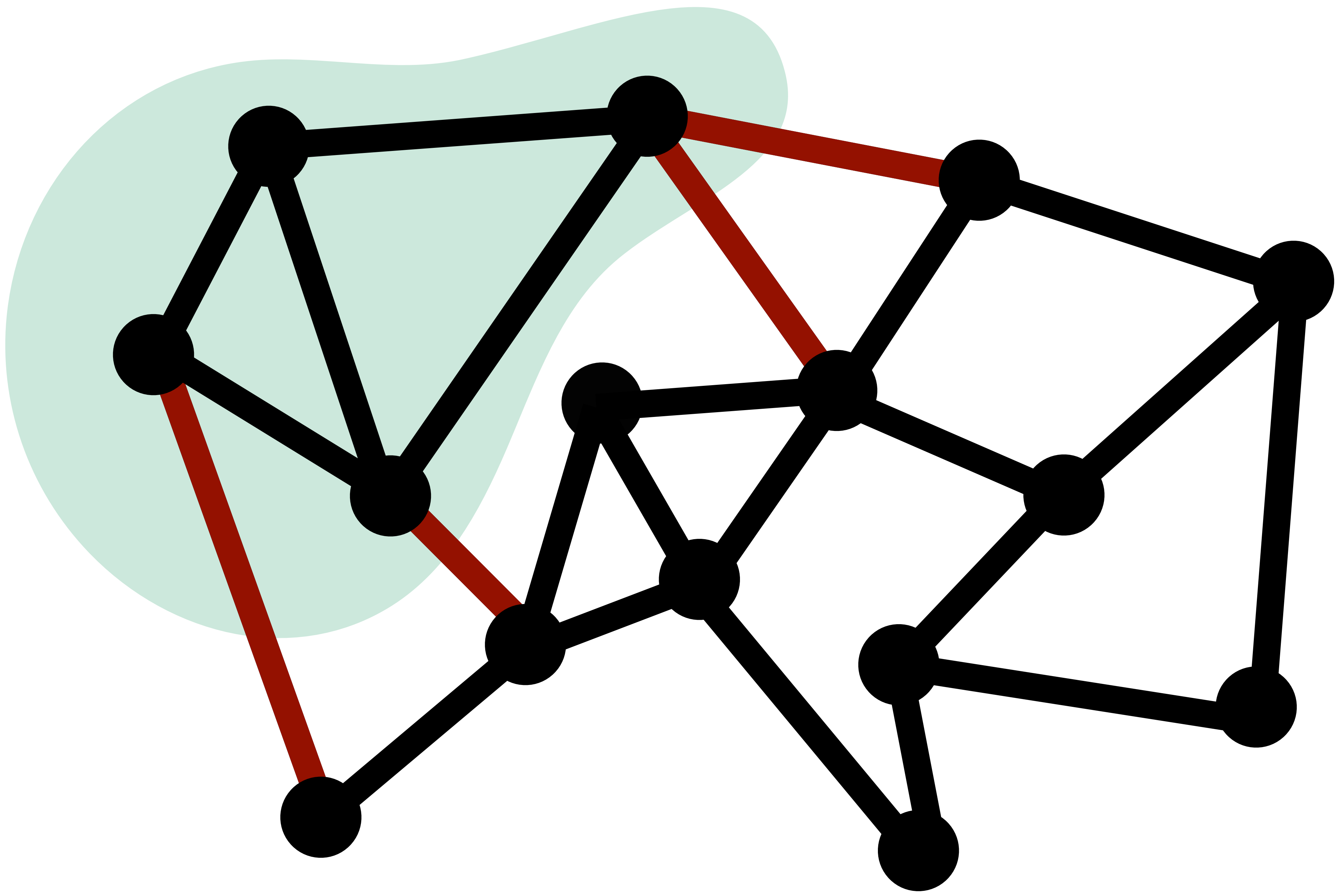


graph $G = (V, E)$

Definition (Edges of Cut S): $\delta(S) := \{(u, v) \in E : u \in S, v \notin S\}$

Papers Overview

Background: Cuts

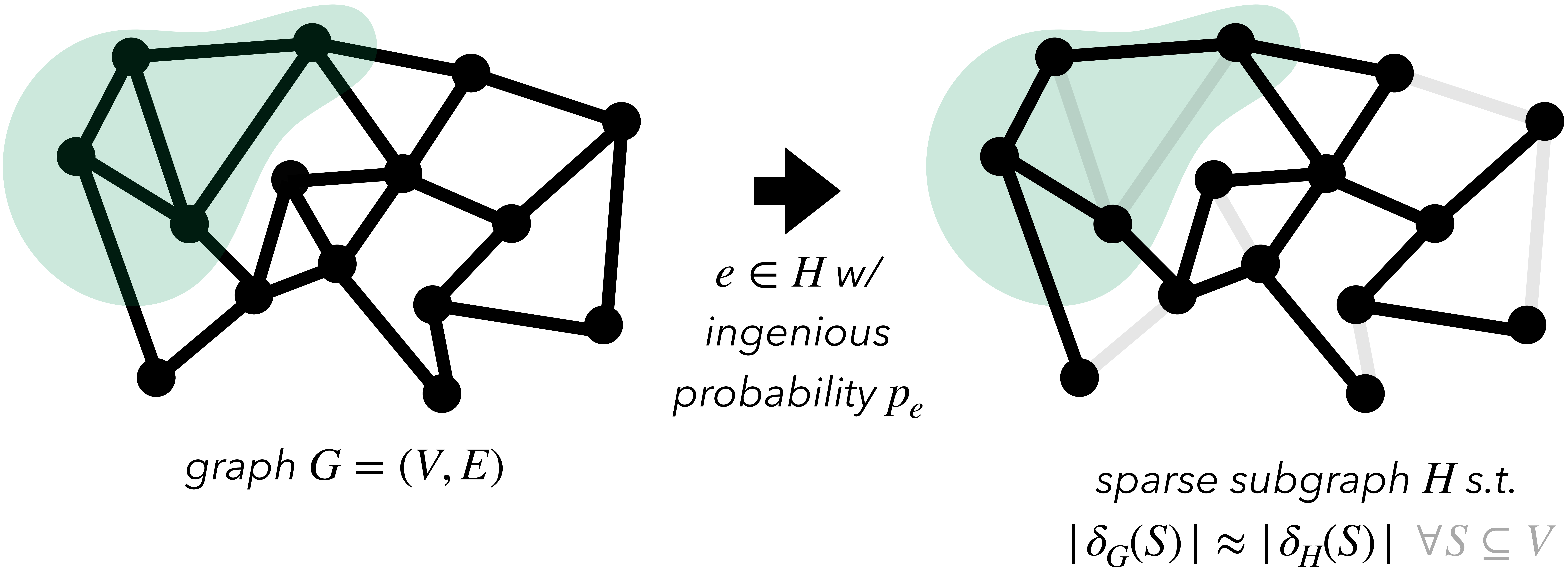


graph $G = (V, E)$

Definition (Cut Size)**:** size of cut S is $|\delta(S)|$

Papers Overview

Paper 5: Sampling-Based Cut Sparsification



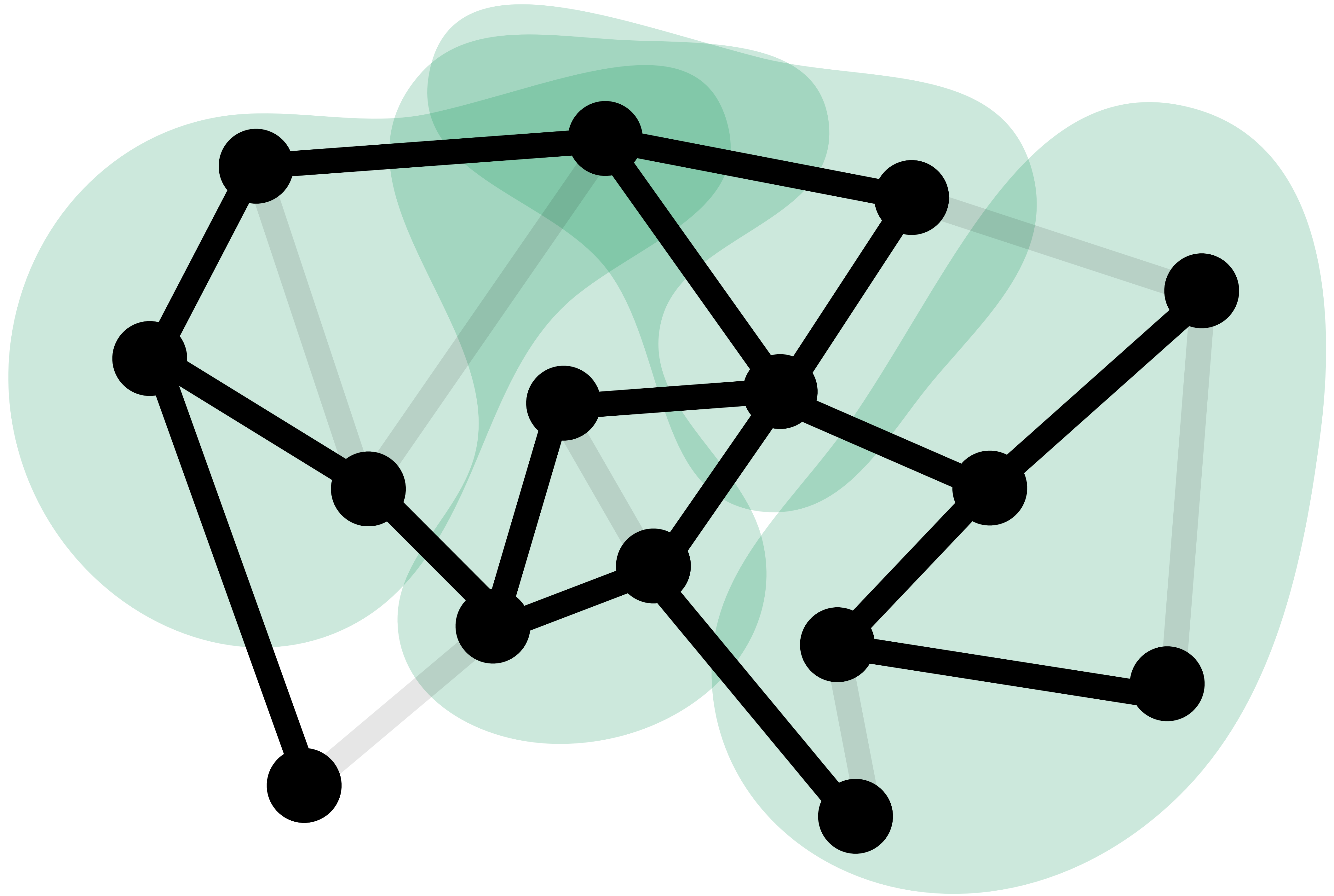
Goal: sparse (edge-weighted) subgraph approximating all cut sizes

Papers Overview

Paper 5: Sampling-Based Cut Sparsification

How this sort of thing is usually argued

- A given cut S has $|\delta_G(S)| \not\approx |\delta_H(S)|$ with tiny probability p
- There are only $k \ll \frac{1}{p}$ cuts
- By union bound all cuts S satisfy $|\delta_G(S)| \approx |\delta_H(S)|$ with high prob.

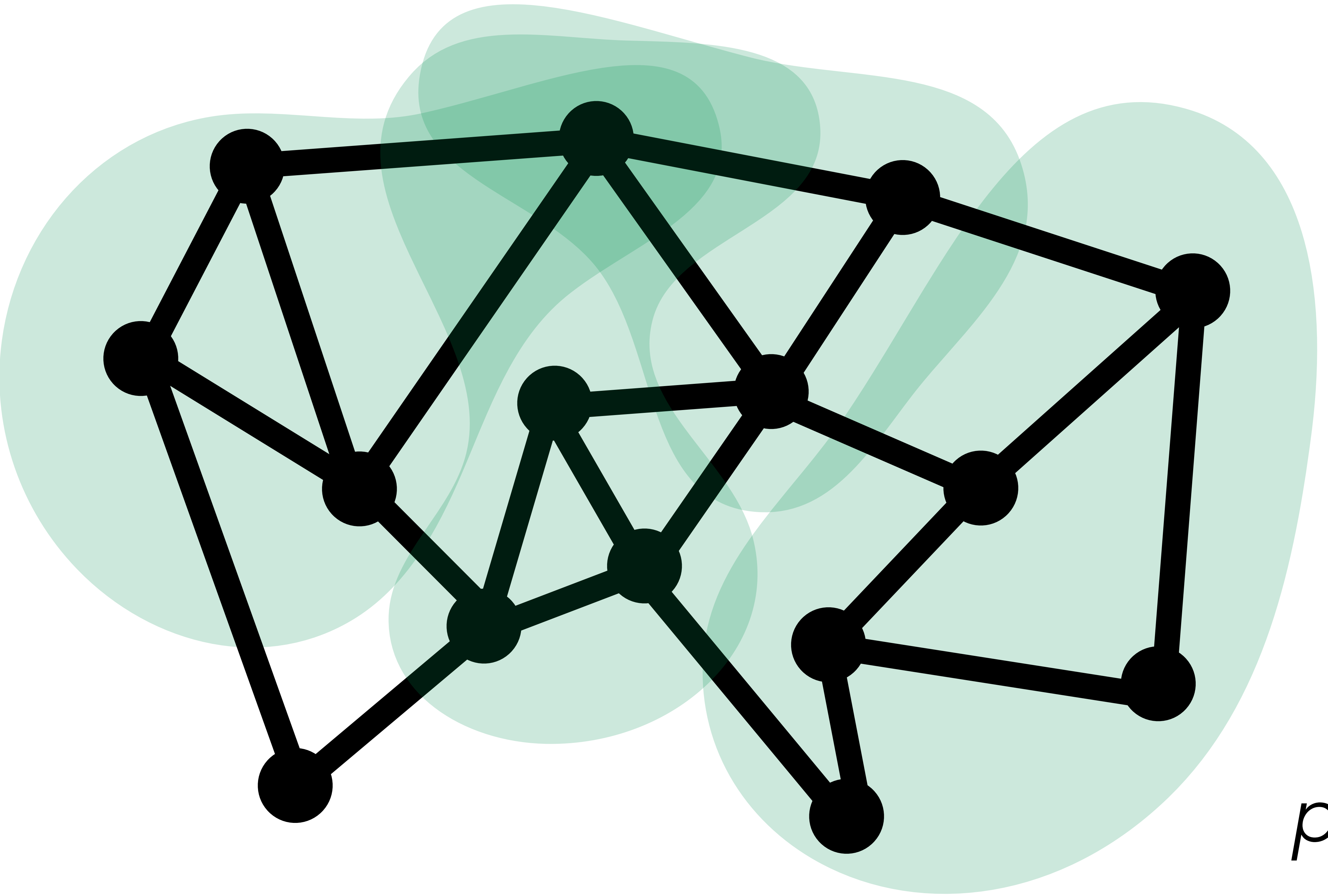


sparse subgraph H *s.t.*

$|\delta_G(S)| \approx |\delta_H(S)| \quad \forall S \subseteq V$

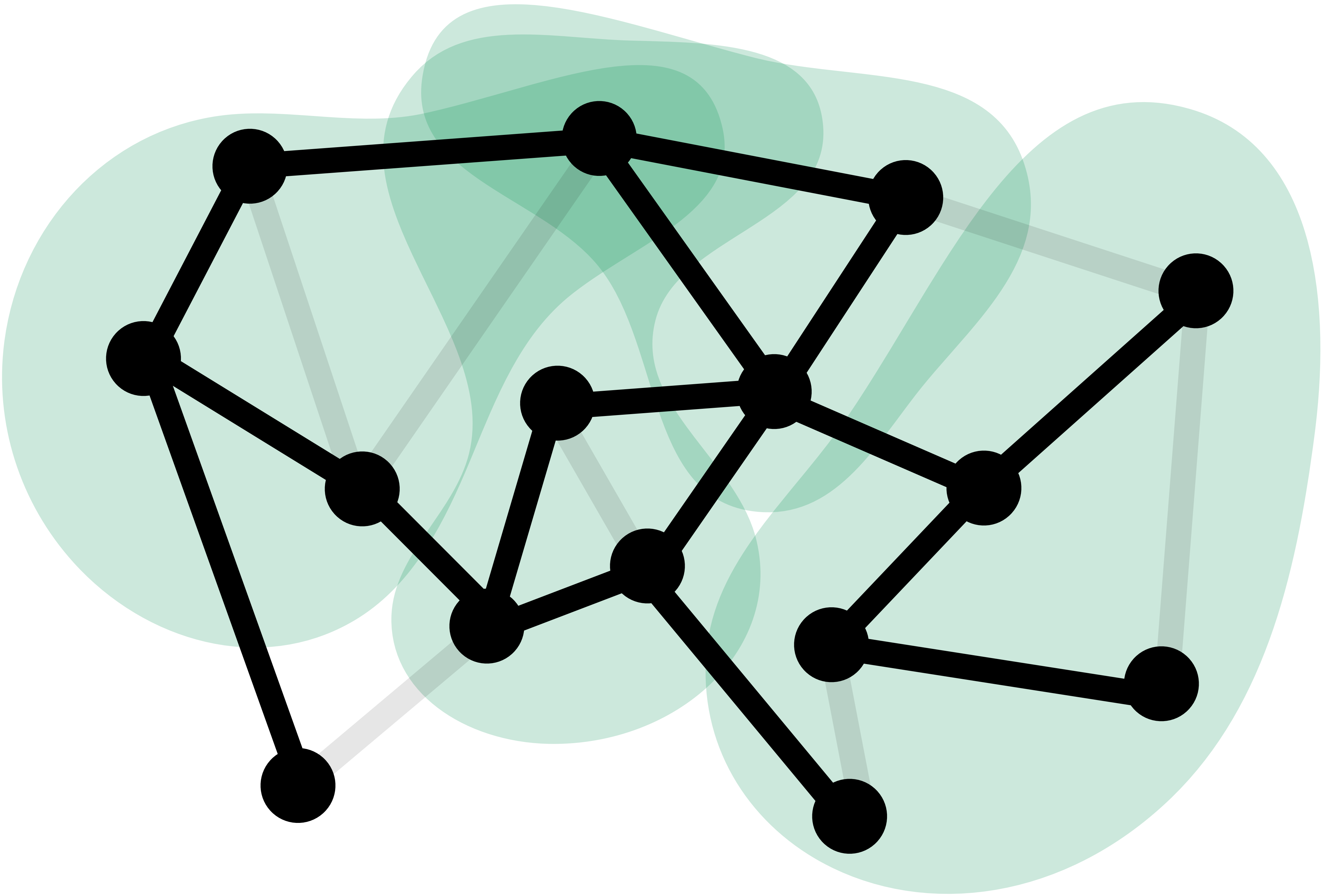
Papers Overview

Paper 5: Sampling-Based Cut Sparsification



graph $G = (V, E)$

$e \in H$ w/ (ingenious) probability p_e



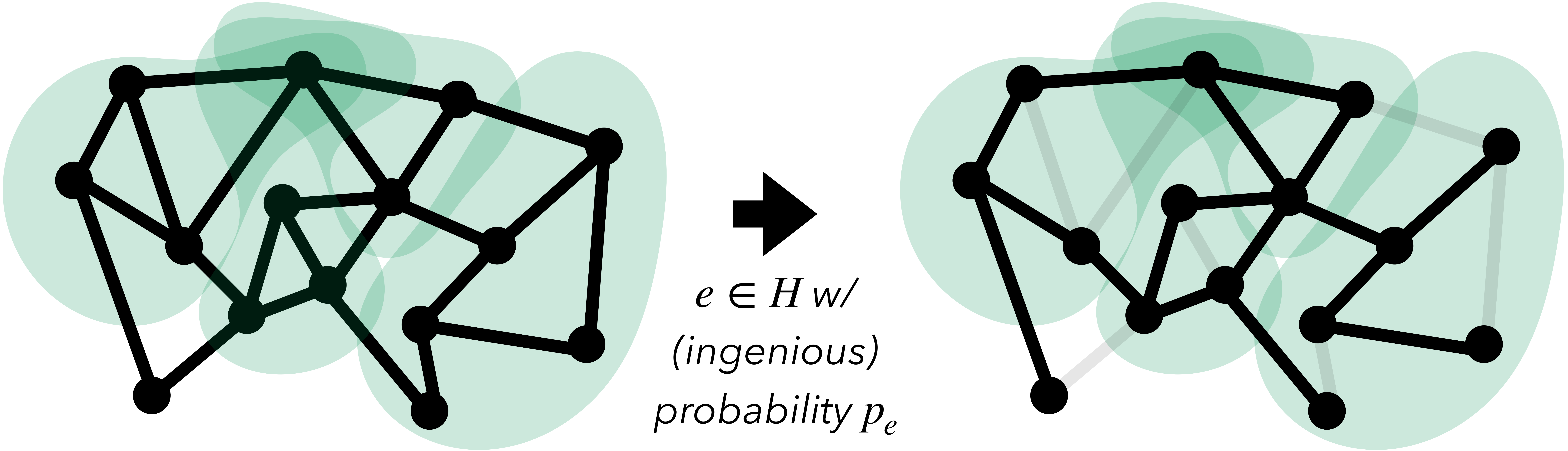
sparse subgraph H s.t.

$|\delta_G(S)| \approx |\delta_H(S)| \quad \forall S \subseteq V$

Problem: $O(2^n)$ cuts, need absurdly good chance of $|\delta_G(S)| \approx |\delta_H(S)|$ for each S

Papers Overview

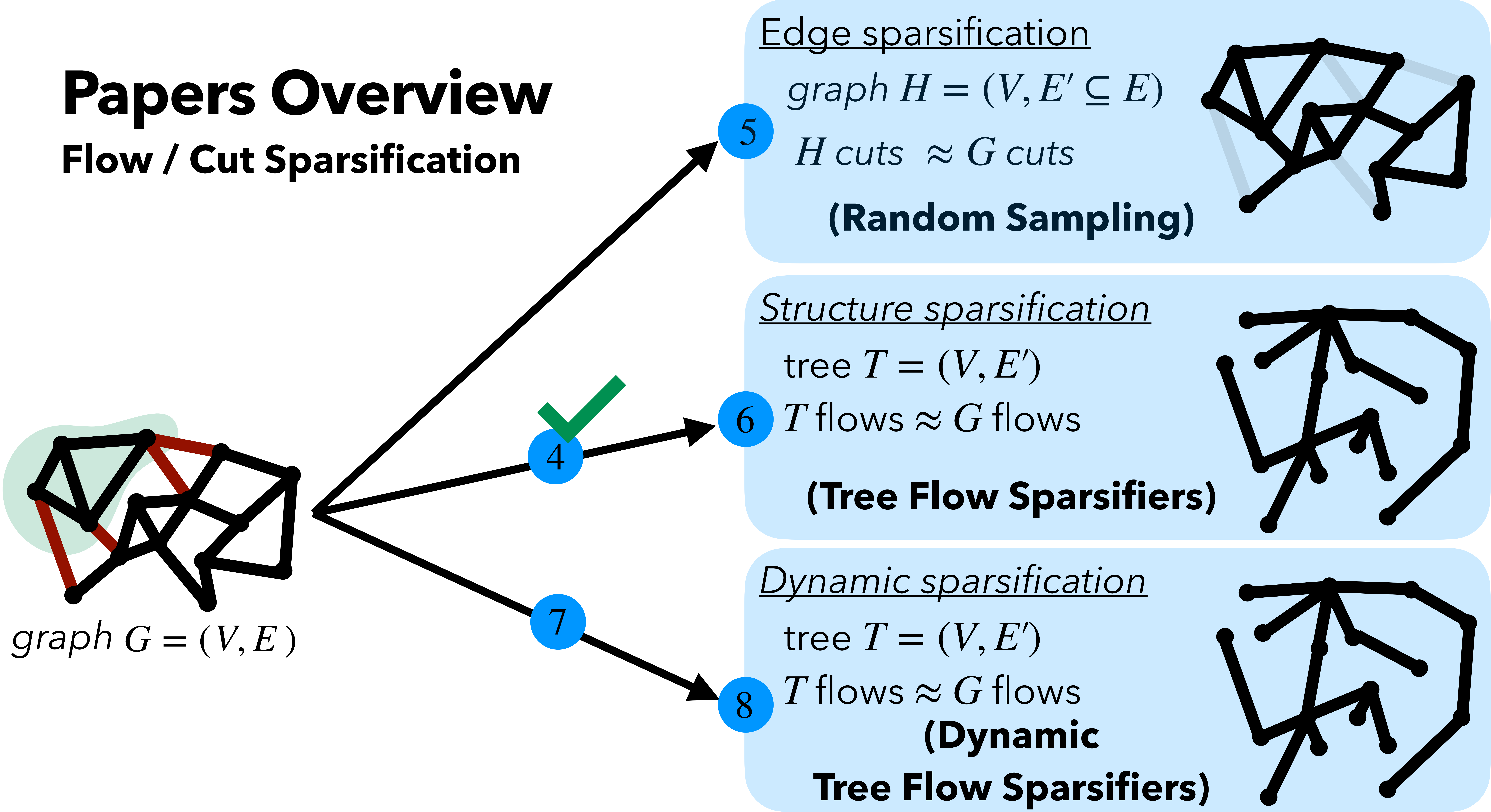
Paper 5: Sampling-Based Cut Sparsification



Theorem: for any $\epsilon > 0$ can choose p_e so with high probability so H

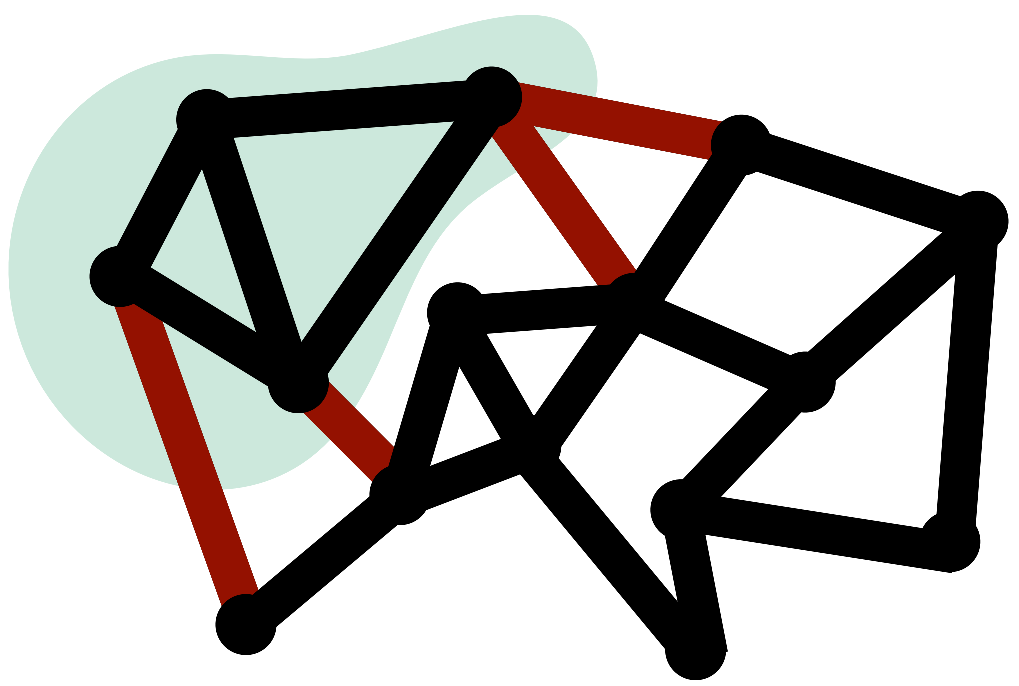
1. has $O(n \log n/\epsilon^2)$ edges
2. preserves all cuts up to $(1 + \epsilon)$ multiplicative factor

(and applications)


Papers Overview
Flow / Cut Sparsification
graph G = (V, E)
4
5
6
7
8
Edge sparsification
graph H = (V, E′ ⊆ E)
H cuts ≈ G cuts
(Random Sampling)
Structure sparsification
tree T = (V, E′)
T flows ≈ G flows
(Tree Flow Sparsifiers)
Dynamic sparsification
tree T = (V, E′)
T flows ≈ G flows
(Dynamic
Tree Flow Sparsifiers)

Papers Overview

Flow / Cut Sparsification



graph $G = (V, E)$

4 ✓

7

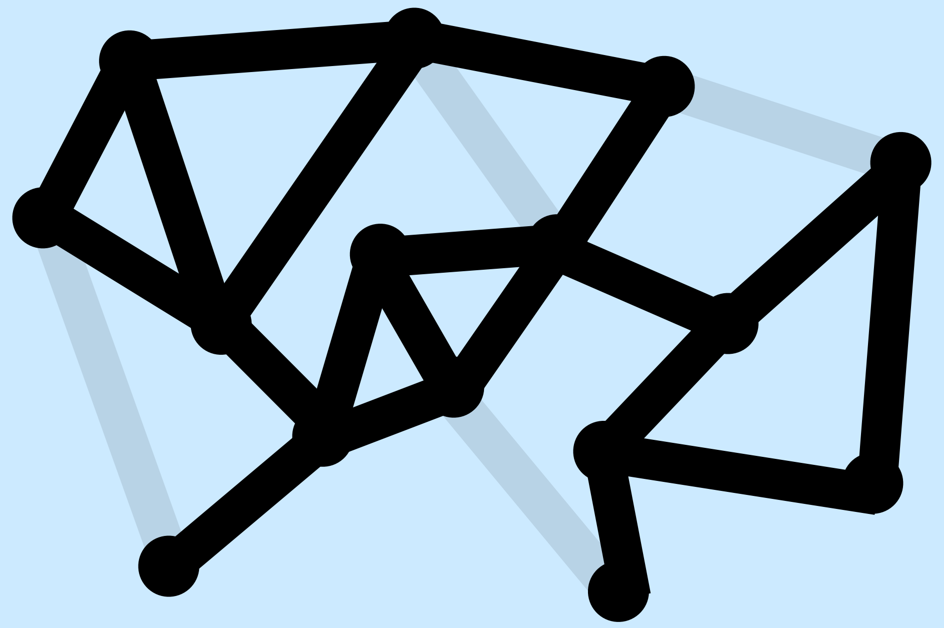
5 ✓

<u>Edge sparsification</u>

graph $H = (V, E' \subseteq E)$

H cuts $\approx G$ cuts

(Random Sampling)



6

<u>*Structure sparsification*</u>

tree $T = (V, E')$

T flows $\approx G$ flows

(Tree Flow Sparsifiers)

8

<u>*Dynamic sparsification*</u>

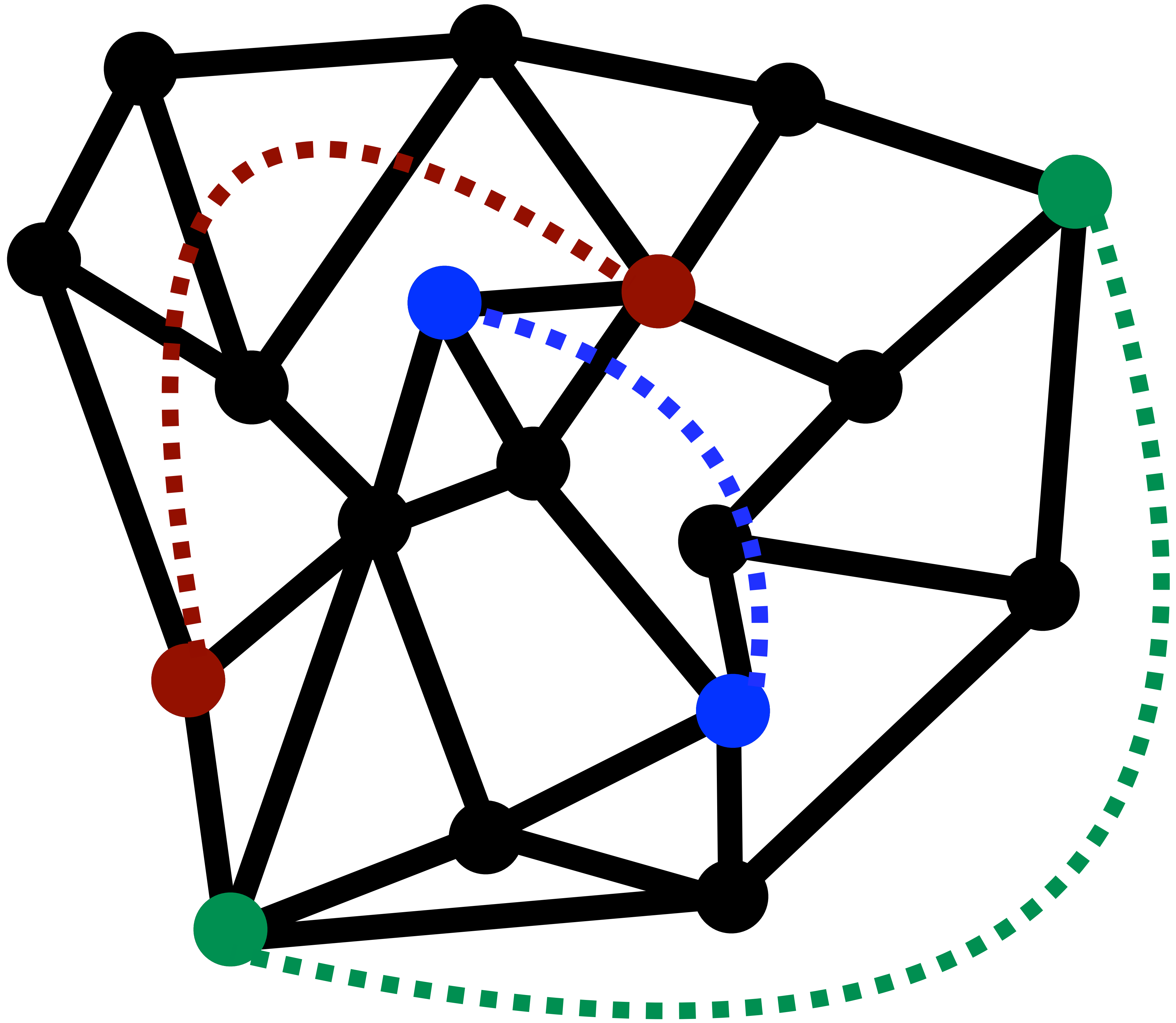
tree $T = (V, E')$

T flows $\approx G$ flows

(Dynamic Tree Flow Sparsifiers)

Papers Overview

Background: (Multi-Commodity) Flows



Goal: some way of formalizing how to send information in a network

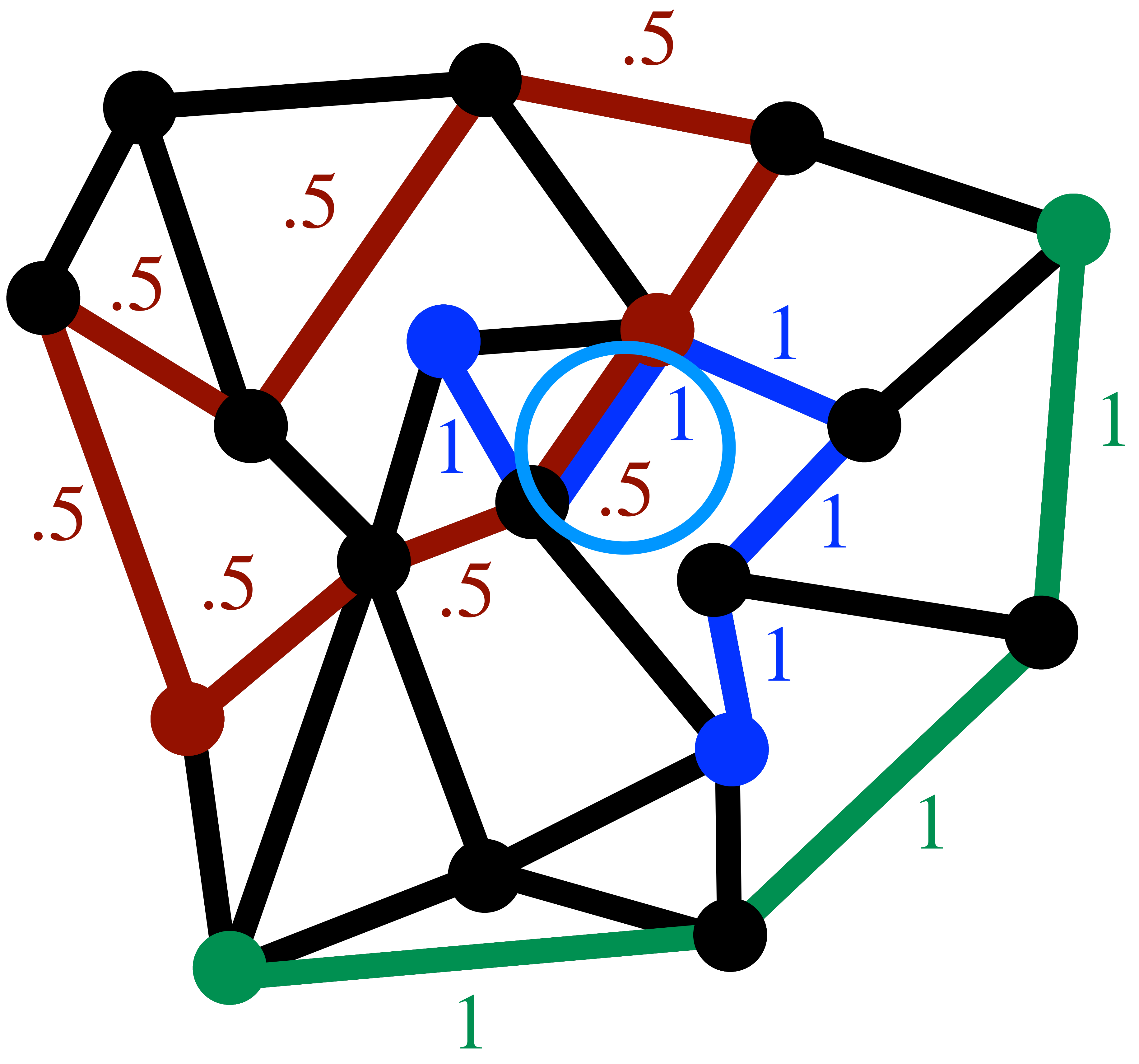
Papers Overview

Background: (Multi-Commodity) Flows

- **Given:**
  - Graph $G = (V, E)$
  - Vertex "demand" pairs $\{(s_i, t_i)\}_i$
- **Goal:**
  - Assign "flow values" f_P to each $s_i - t_i$ path P so each pair sends 1 flow
  - Minimize congestion $:= \max_e \sum_{P \ni e} f_P$

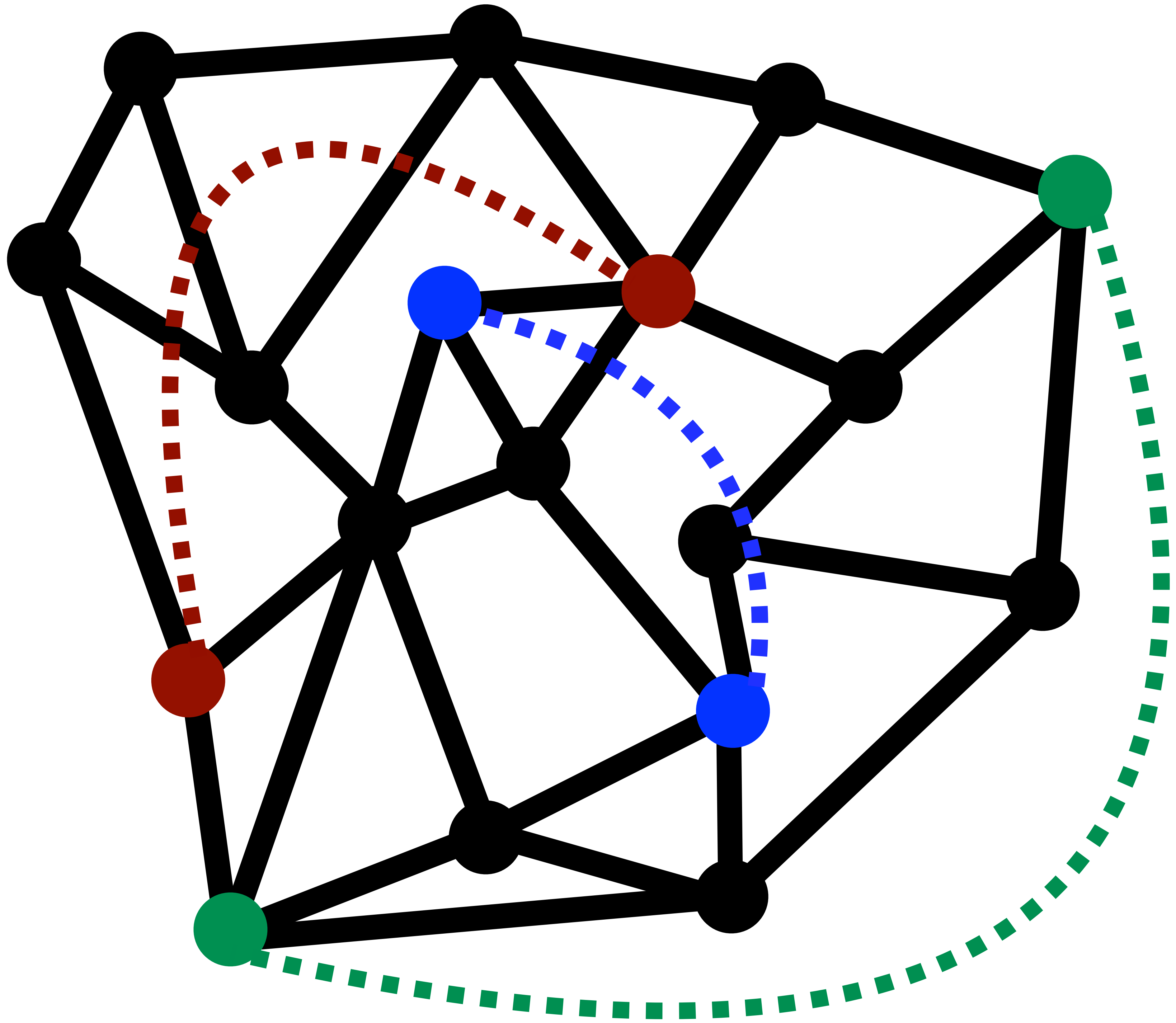
Can solve in poly-time

Optimal Flow ≈ Min Cut



Papers Overview

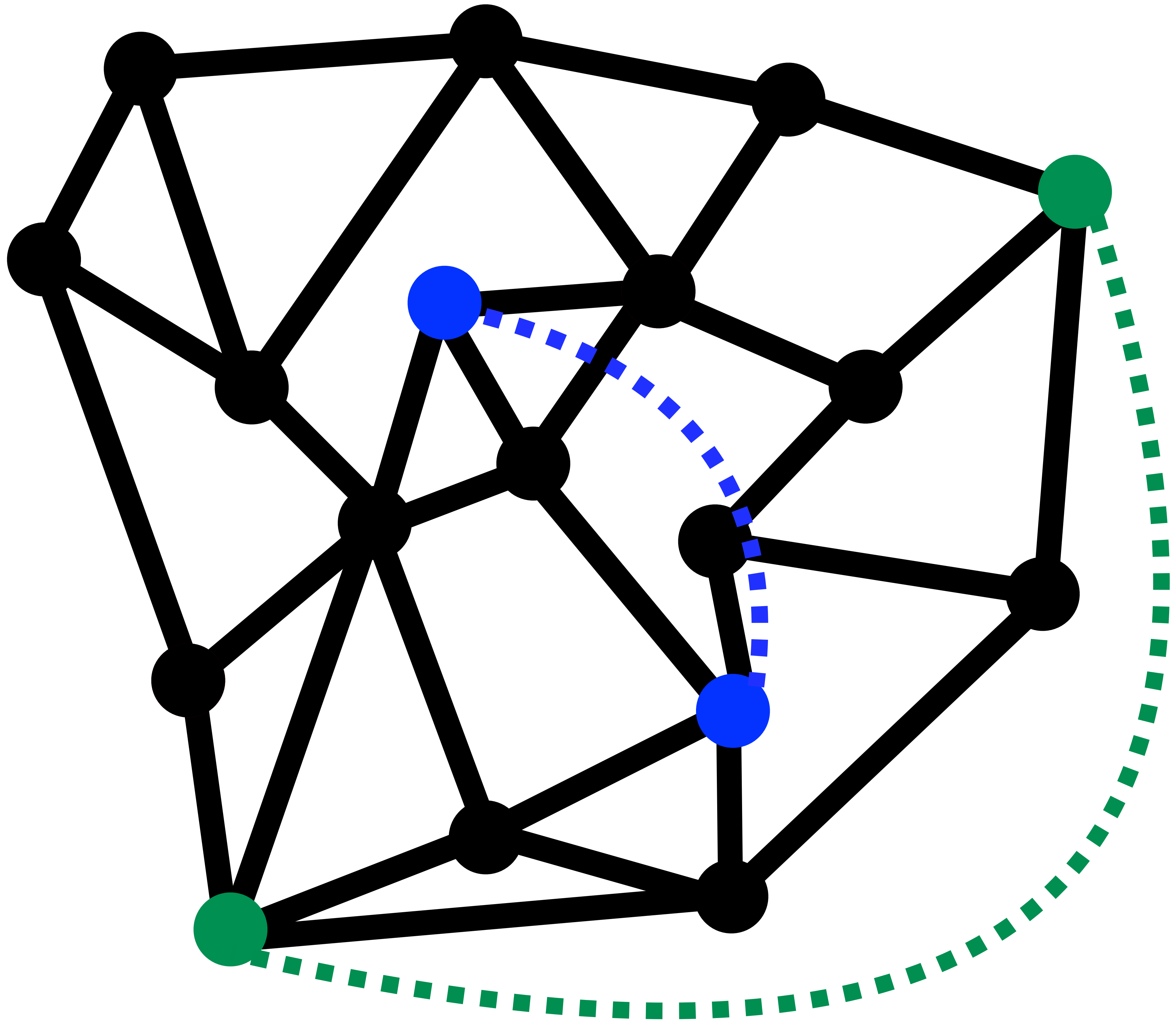
Paper 6: Tree Flow Sparsifiers



Problem: demands change over time, don't want to recompute from scratch

Papers Overview

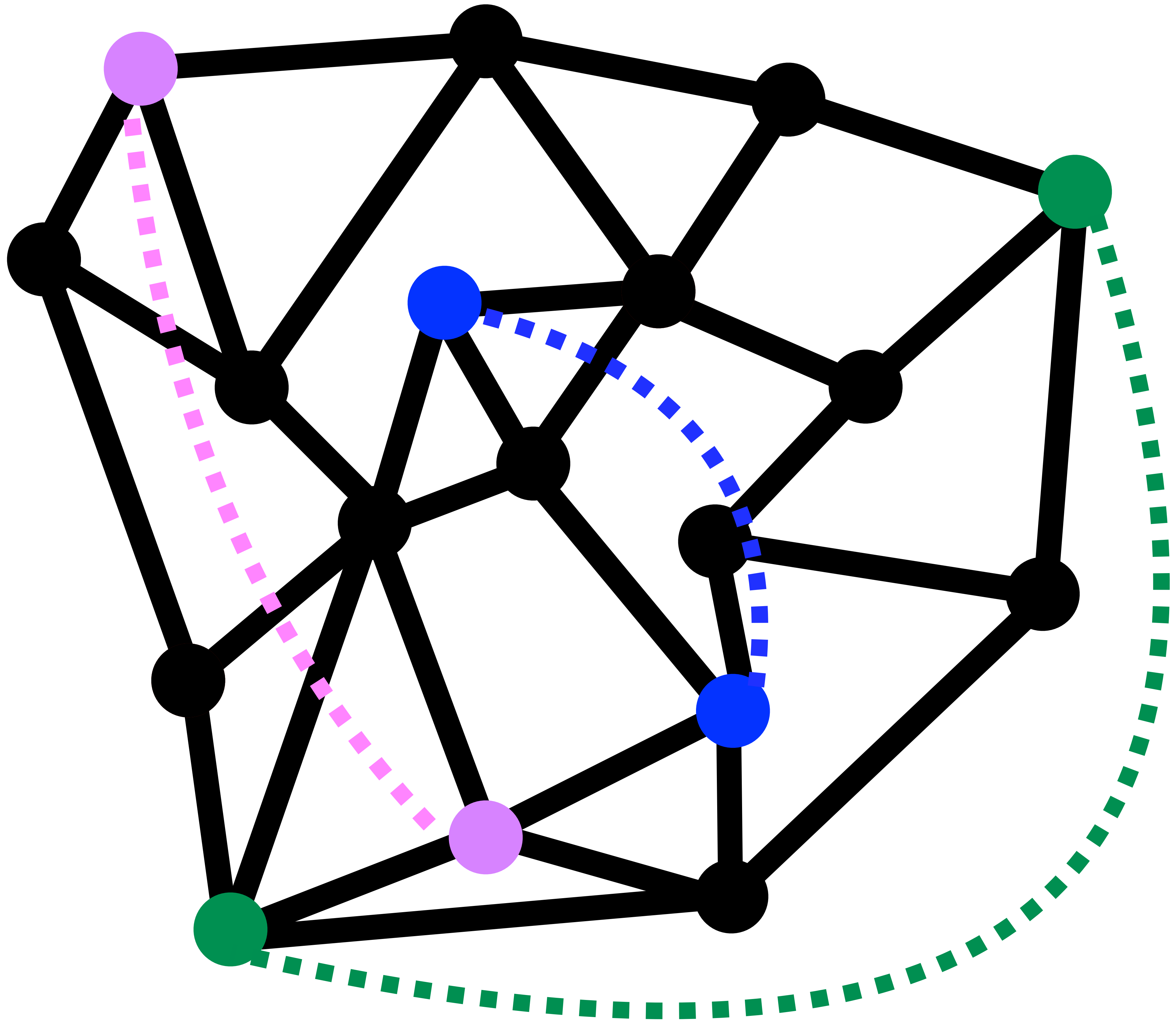
Paper 6: Tree Flow Sparsifiers



Problem: demands change over time, don't want to recompute from scratch

Papers Overview

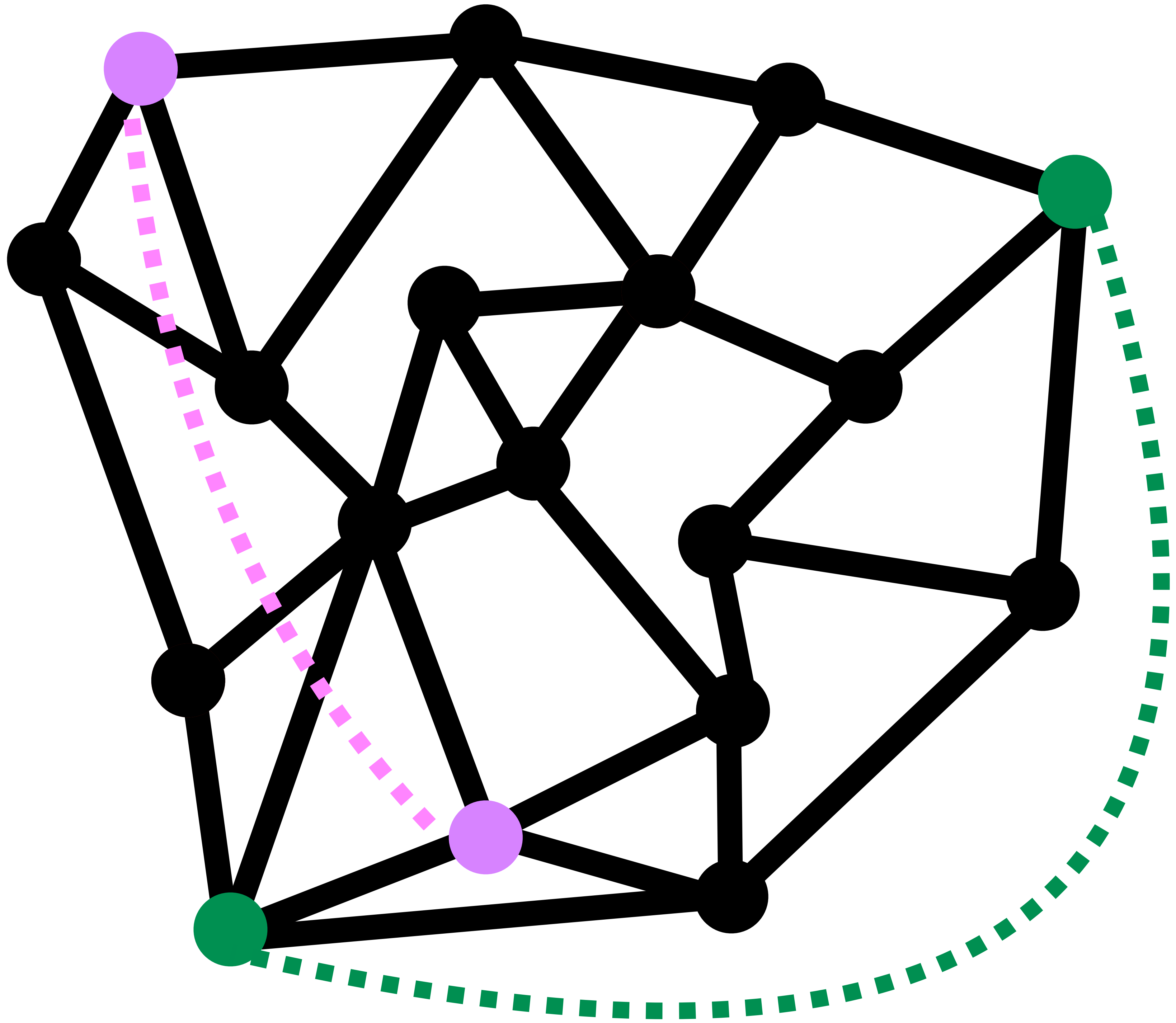
Paper 6: Tree Flow Sparsifiers



Problem: demands change over time, don't want to recompute from scratch

Papers Overview

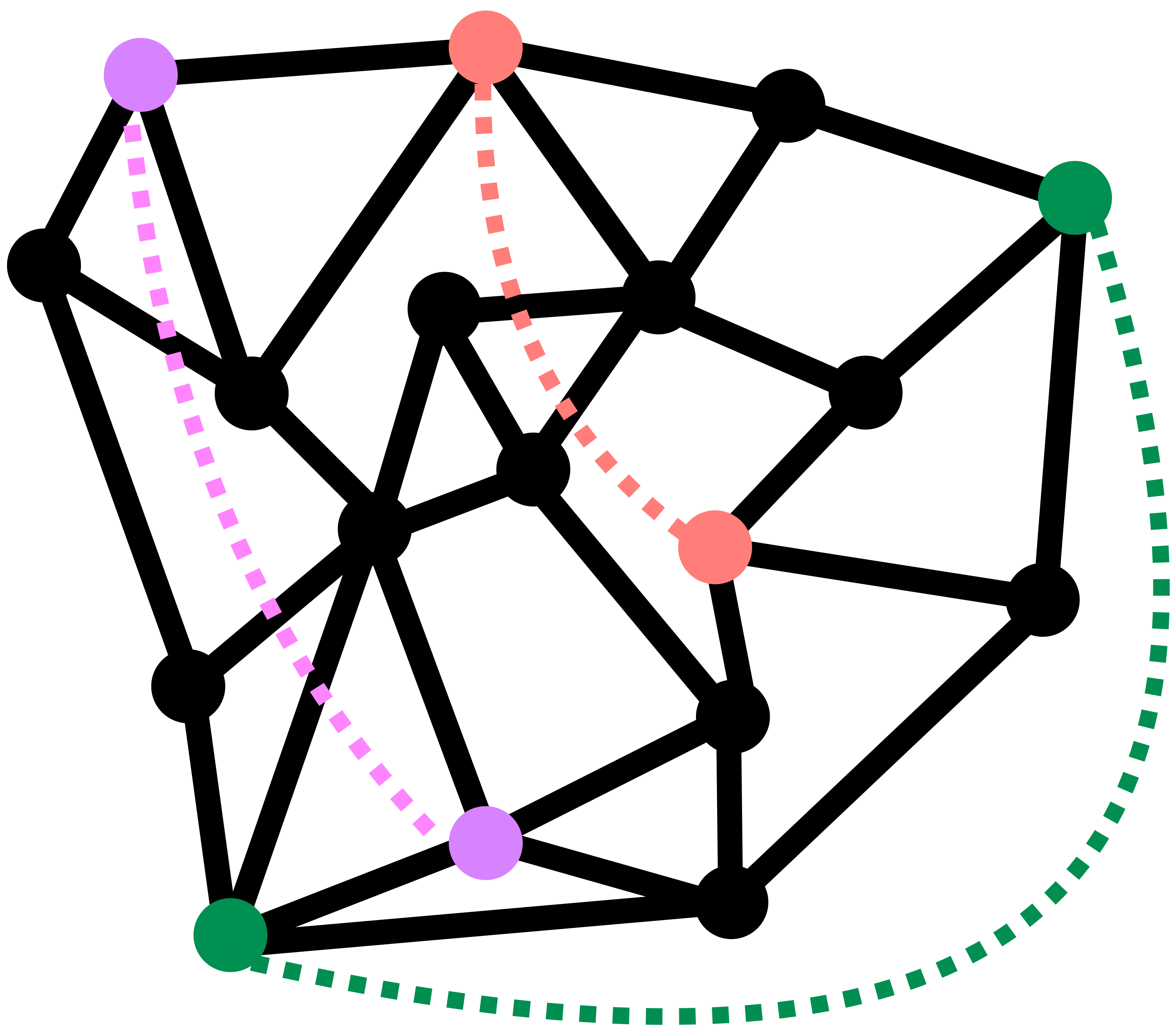
Paper 6: Tree Flow Sparsifiers



Problem: demands change over time, don't want to recompute from scratch

Papers Overview

Paper 6: Tree Flow Sparsifiers

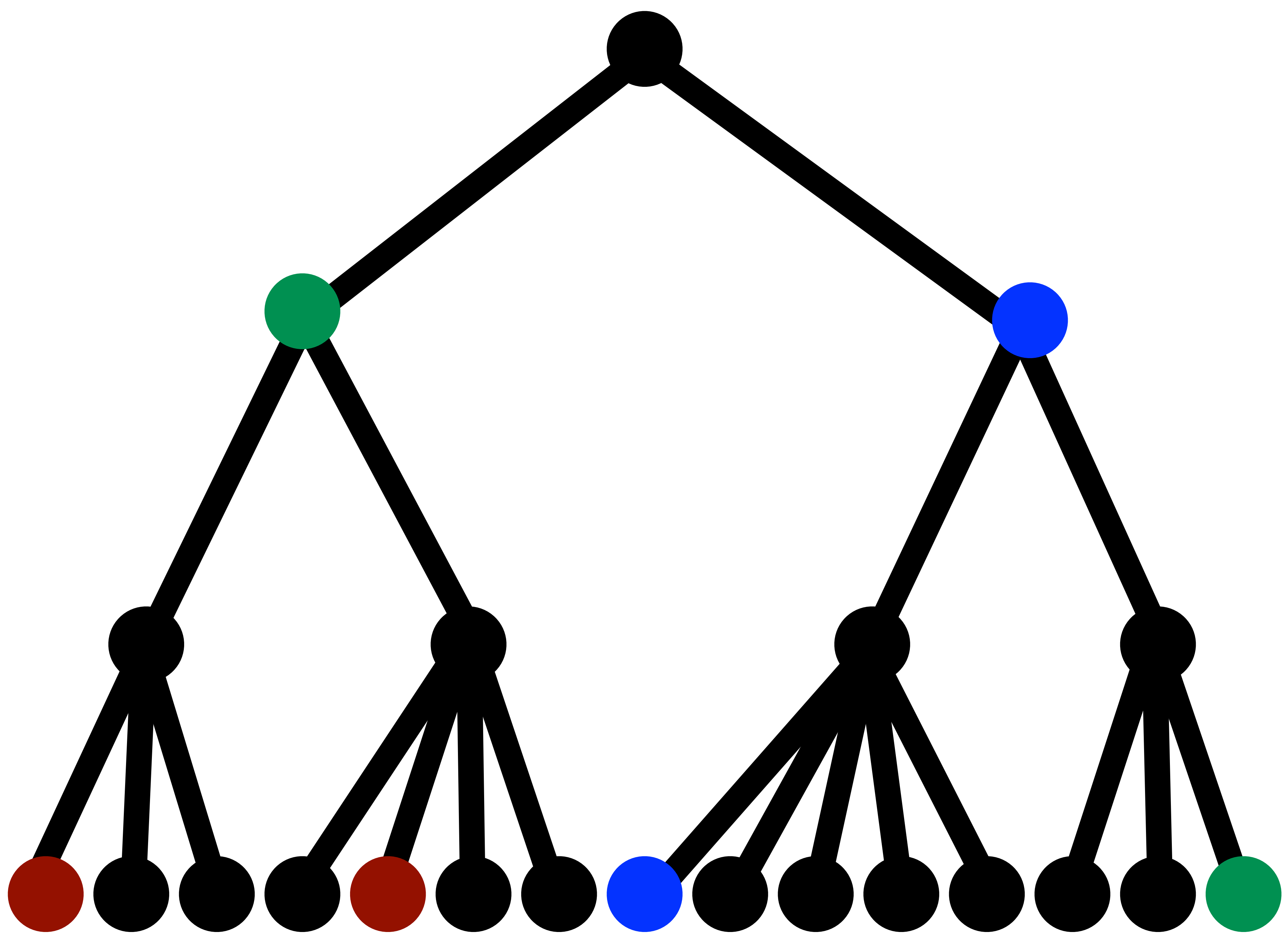


🤔 **What if graph was a tree?**🤔

Problem: demands change over time, don't want to recompute from scratch

Papers Overview

Paper 6: Tree Flow Sparsifiers

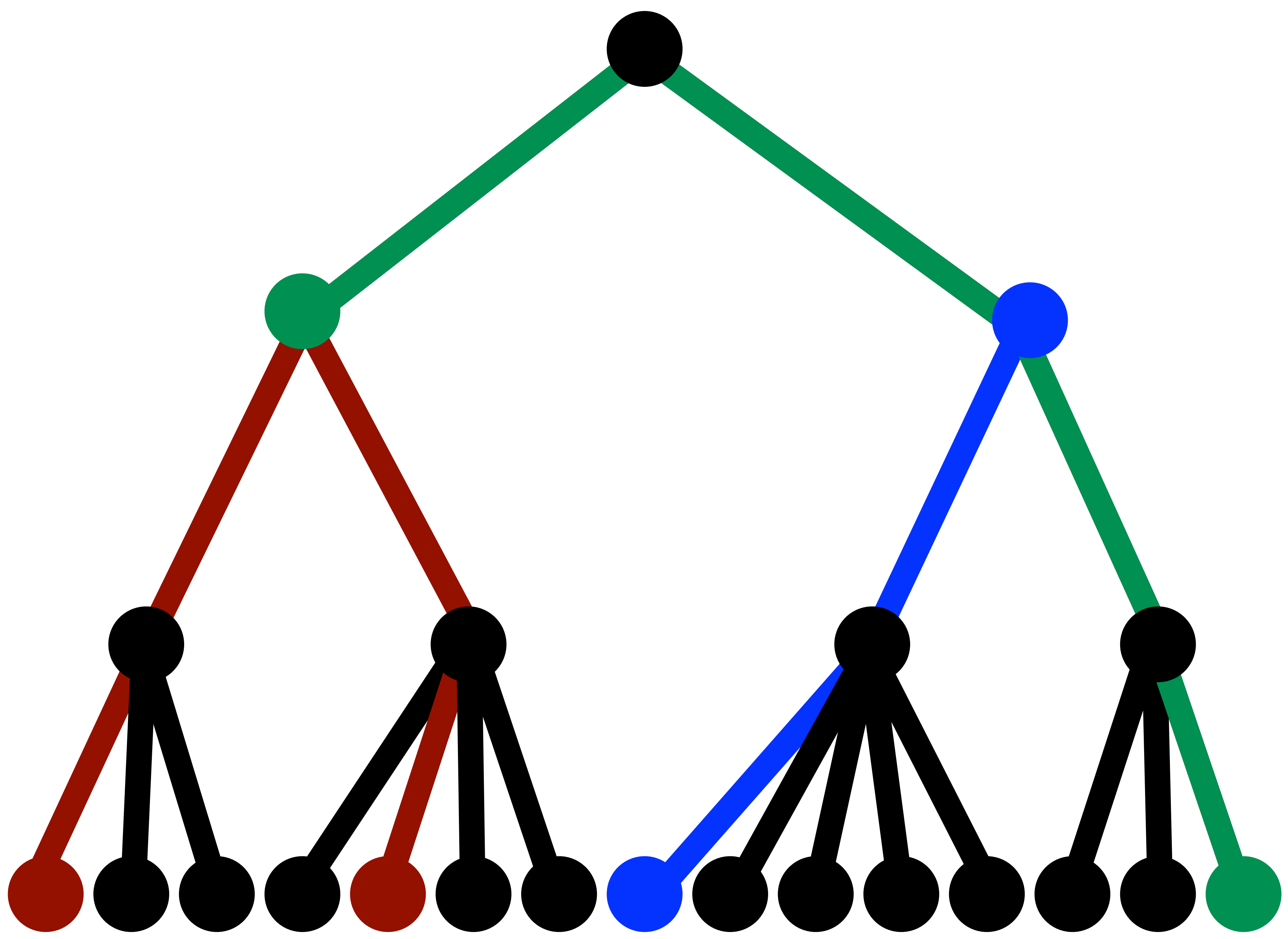


🤔 **What if graph was a tree?**🤔

Problem: demands change over time, don't want to recompute from scratch

Papers Overview

Paper 6: Tree Flow Sparsifiers

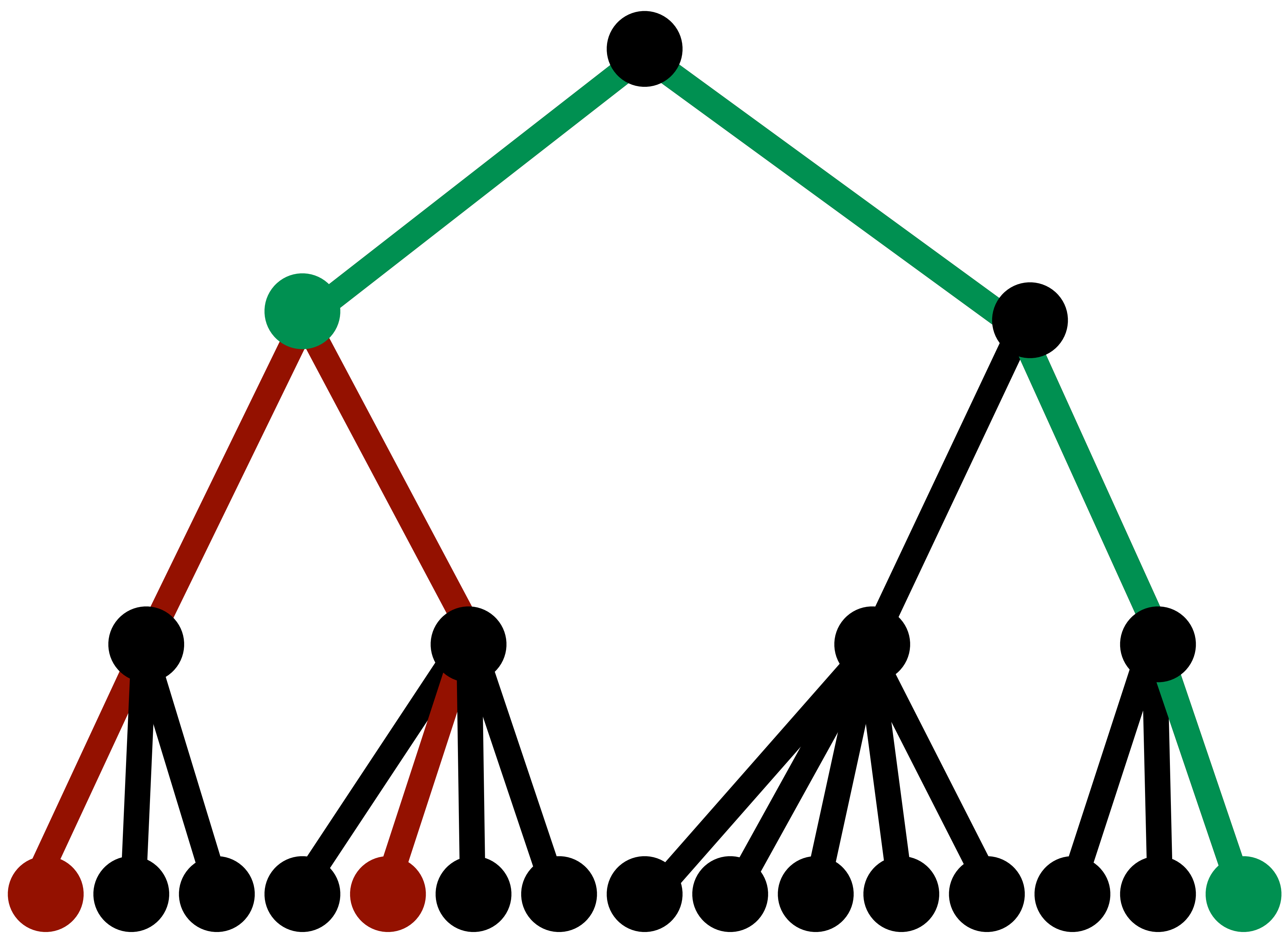


🤔 What if graph was a tree?🤔

Problem: demands change over time, don't want to recompute from scratch

Papers Overview

Paper 6: Tree Flow Sparsifiers

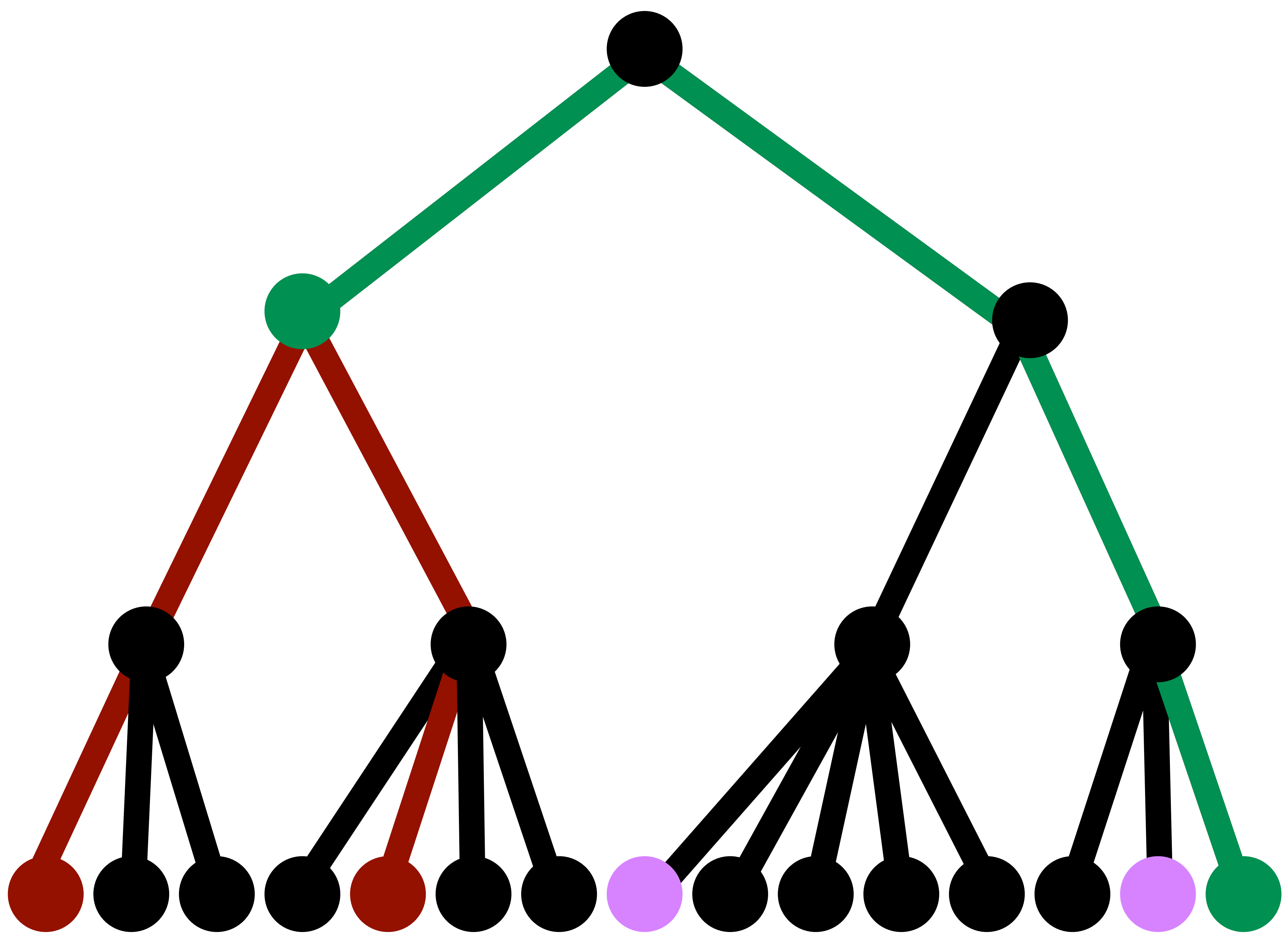


🤔 What if graph was a tree?🤔

Problem: demands change over time, don't want to recompute from scratch

Papers Overview

Paper 6: Tree Flow Sparsifiers

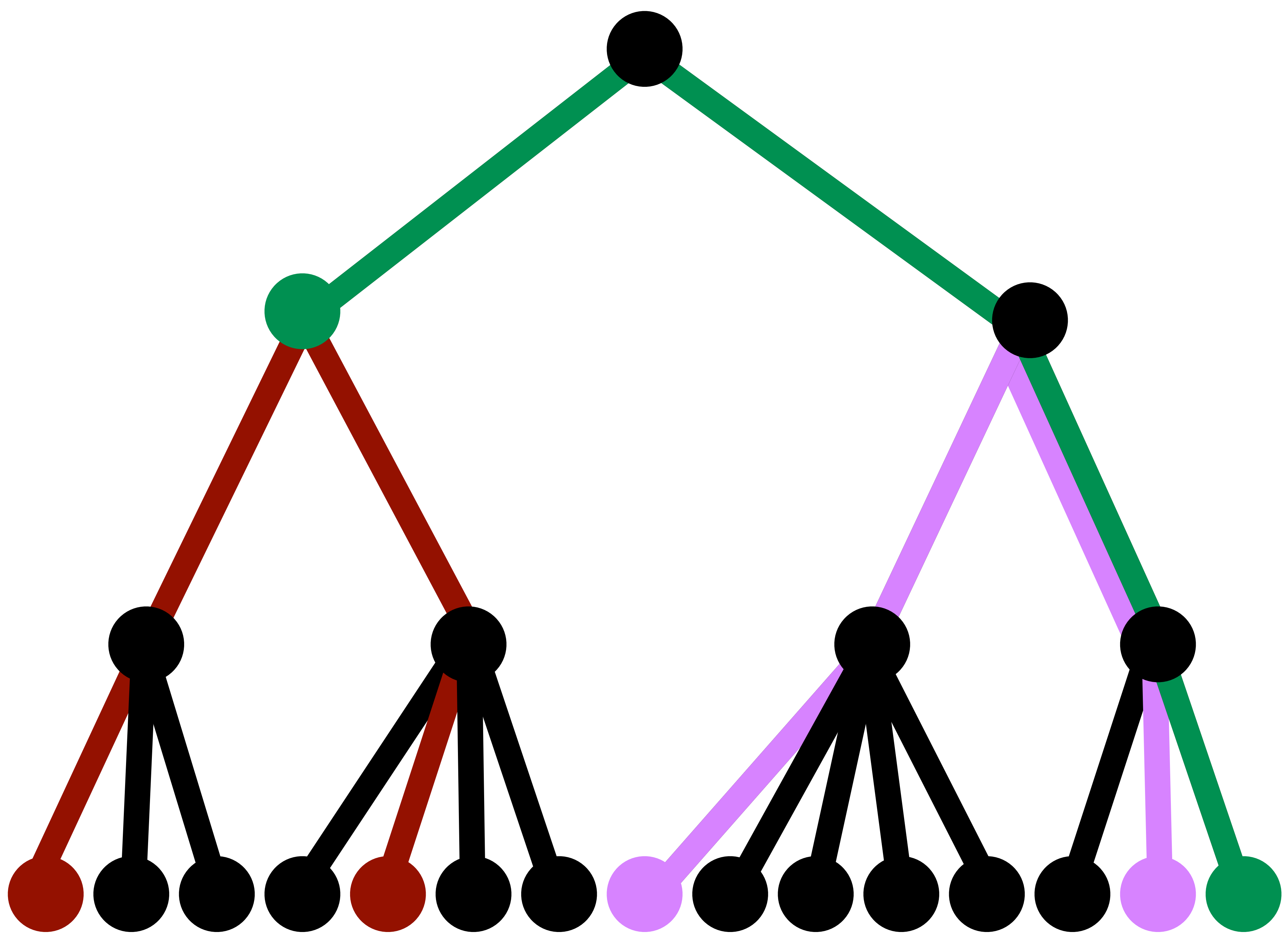


🤔 What if graph was a tree?🤔

Problem: demands change over time, don't want to recompute from scratch

Papers Overview

Paper 6: Tree Flow Sparsifiers

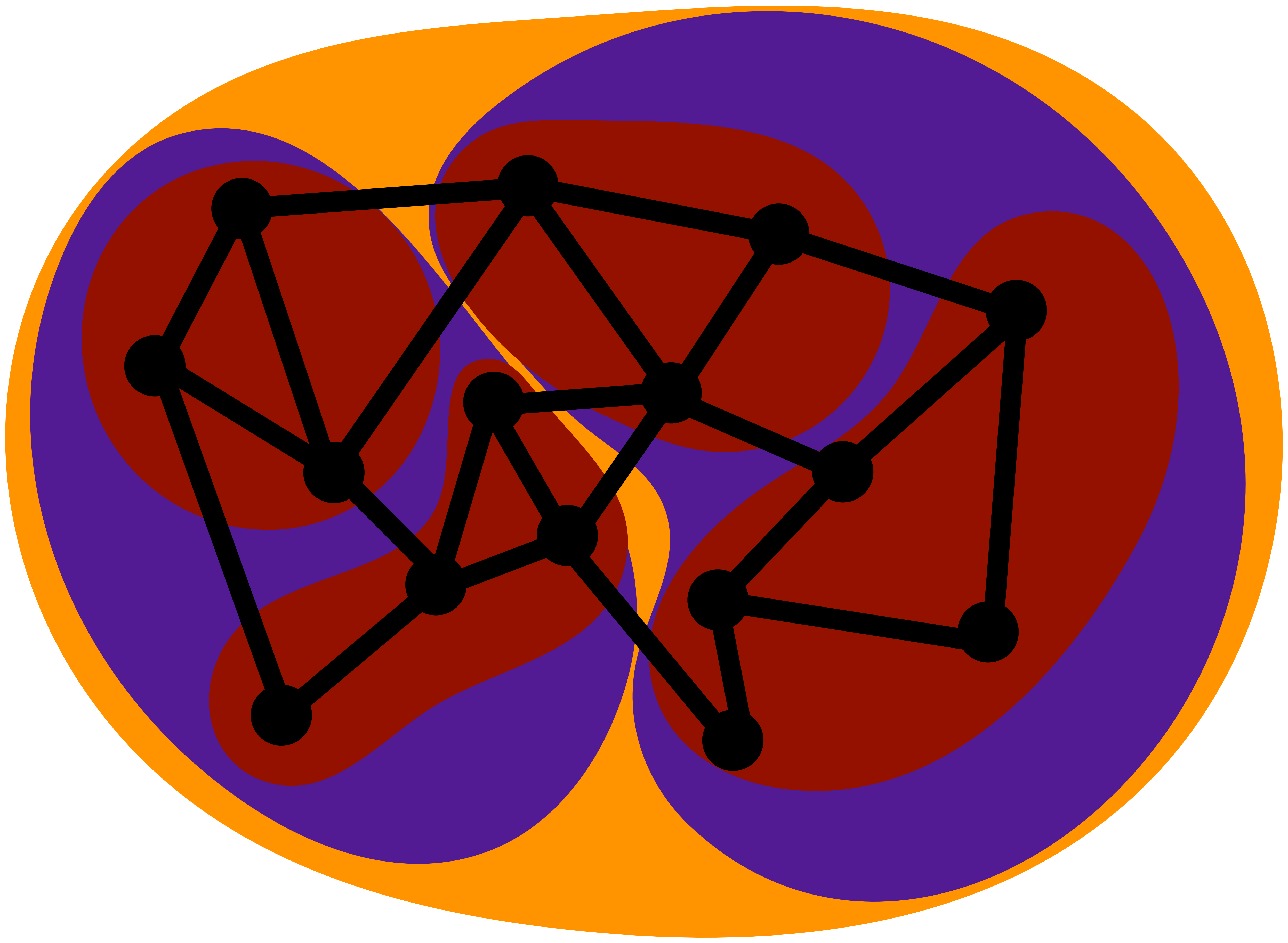


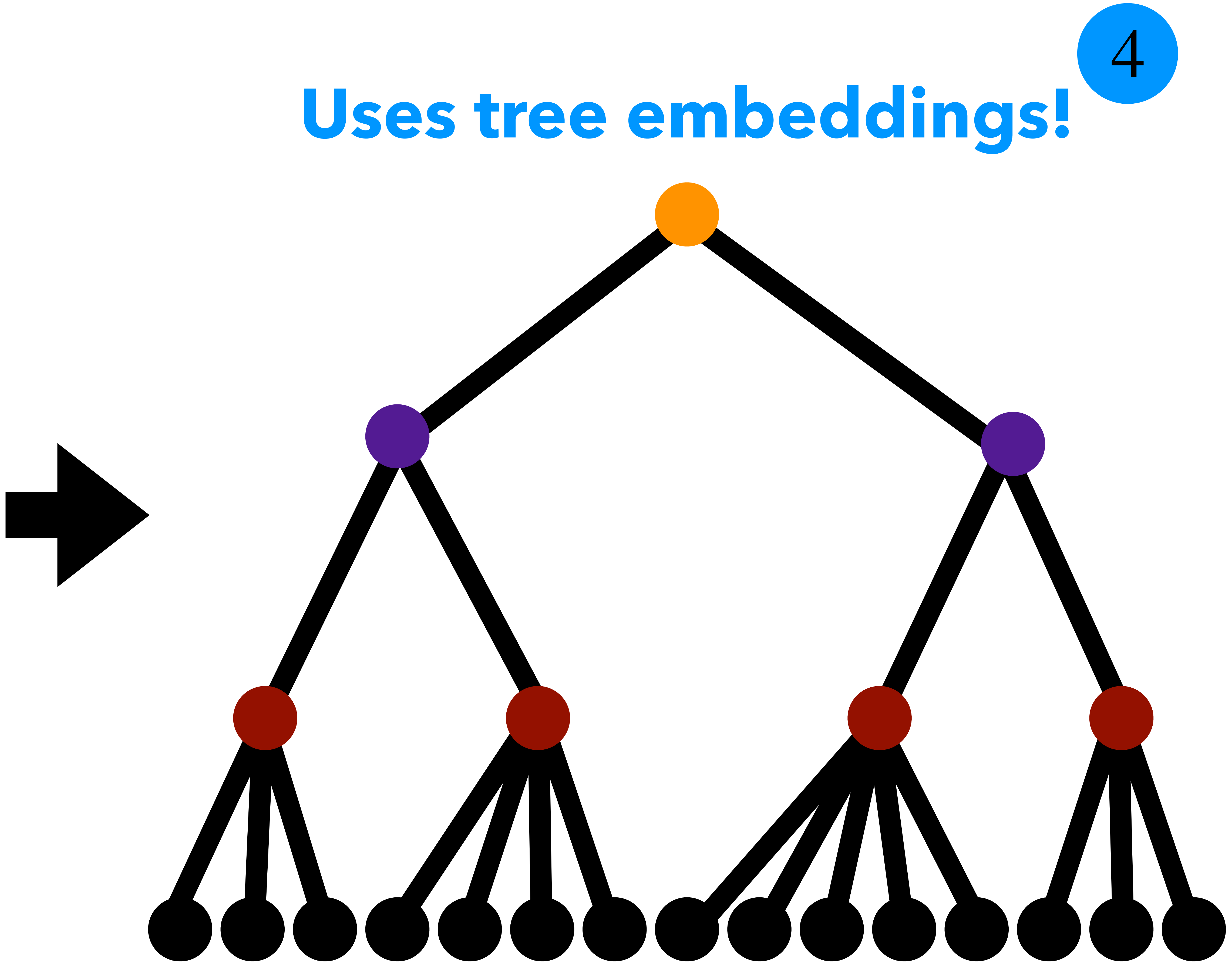
🤔 What if graph was a tree?🤔

Problem: demands change over time, don't want to recompute from scratch

Papers Overview

Paper 6: Tree Flow Sparsifiers



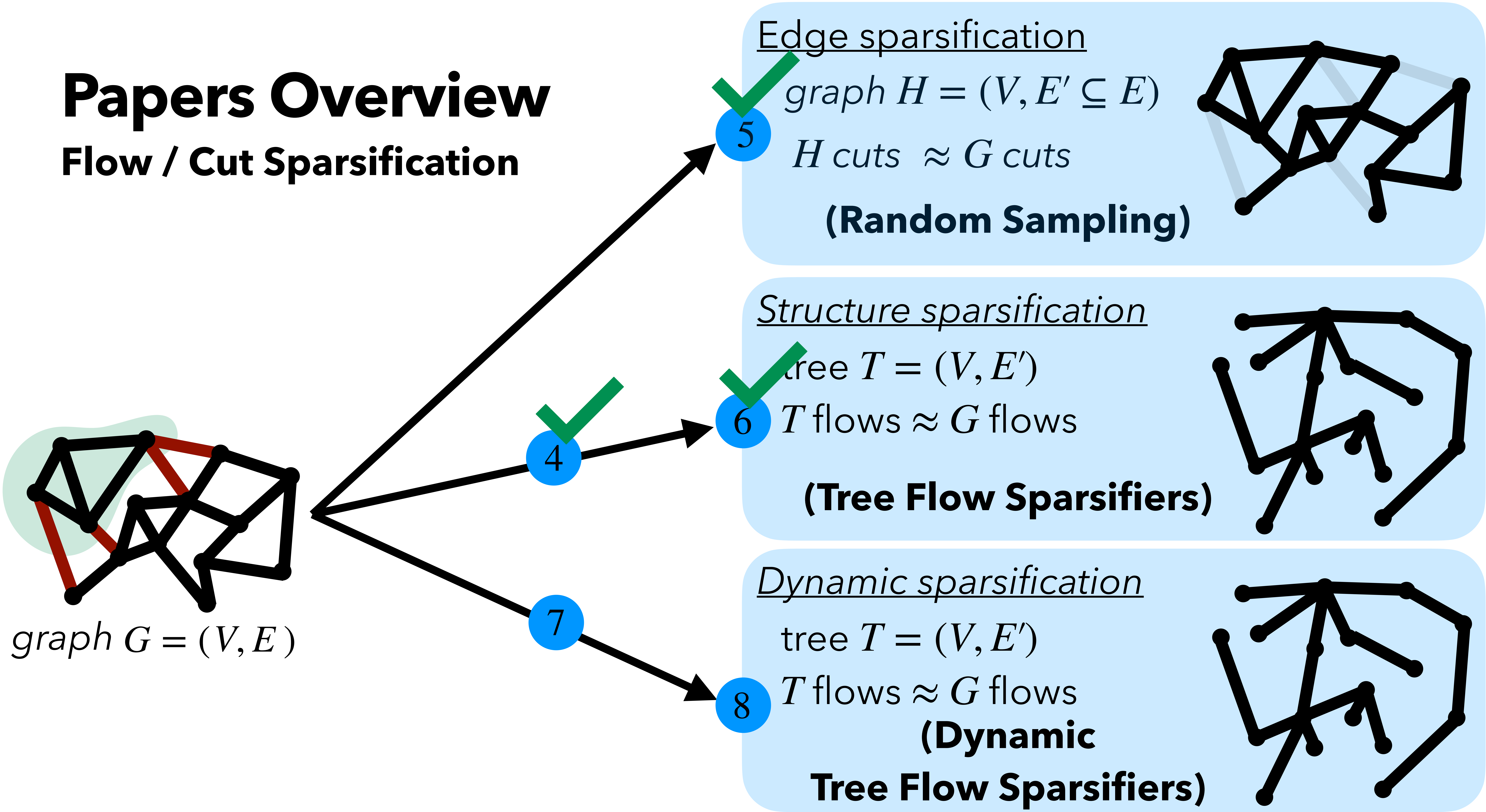


Theorem(informal): can construct a tree approximating "flow structure"

Papers Overview
Flow / Cut Sparsification
graph $G = (V, E)$
4
7
5
Edge sparsification
graph $H = (V, E' \subseteq E)$
H cuts $\approx G$ cuts
(Random Sampling)
6
Structure sparsification
tree $T = (V, E')$
T flows $\approx G$ flows
(Tree Flow Sparsifiers)
8
Dynamic sparsification
tree $T = (V, E')$
T flows $\approx G$ flows
(Dynamic
Tree Flow Sparsifiers)

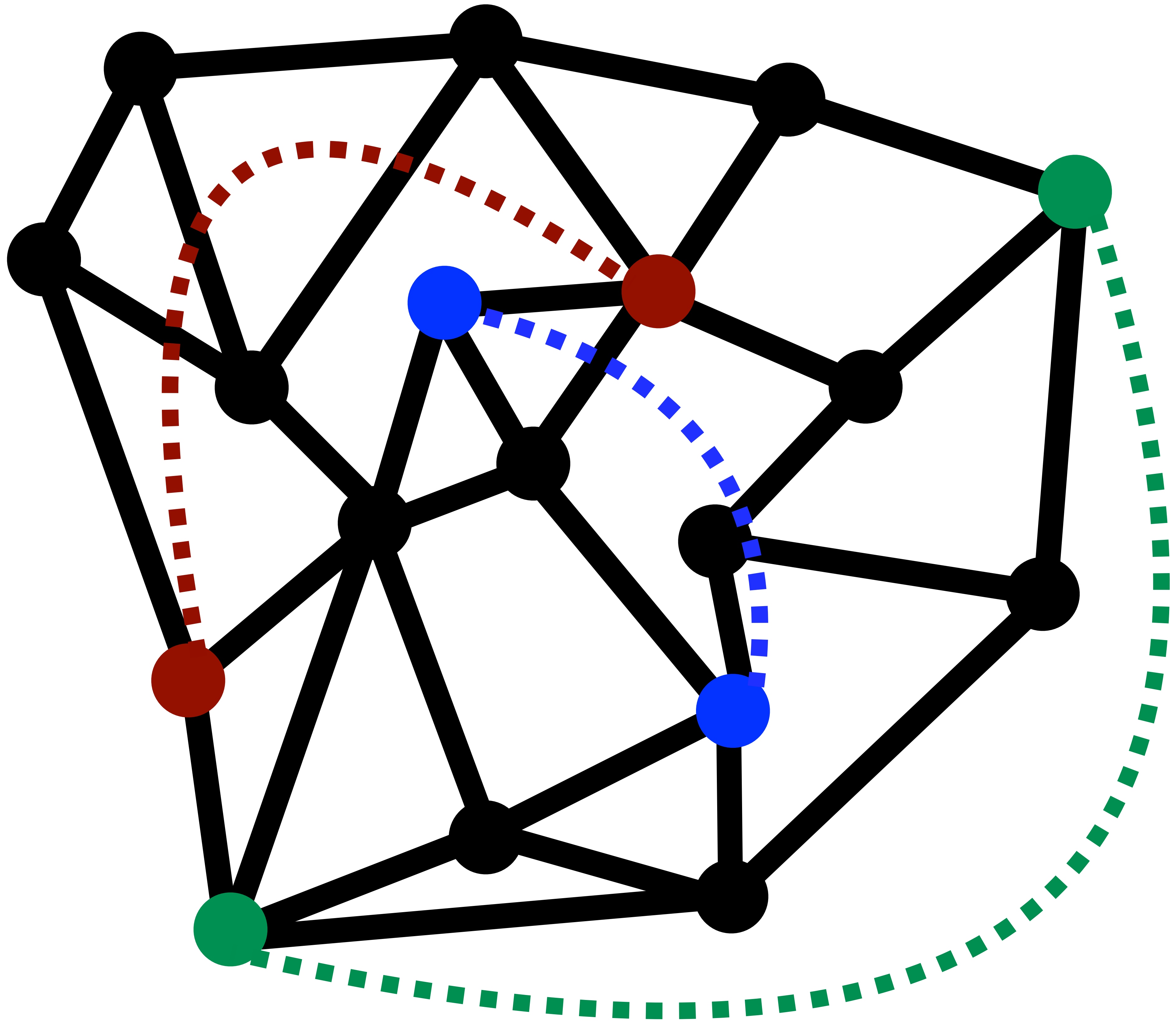
Papers Overview

Flow / Cut Sparsification



Papers Overview

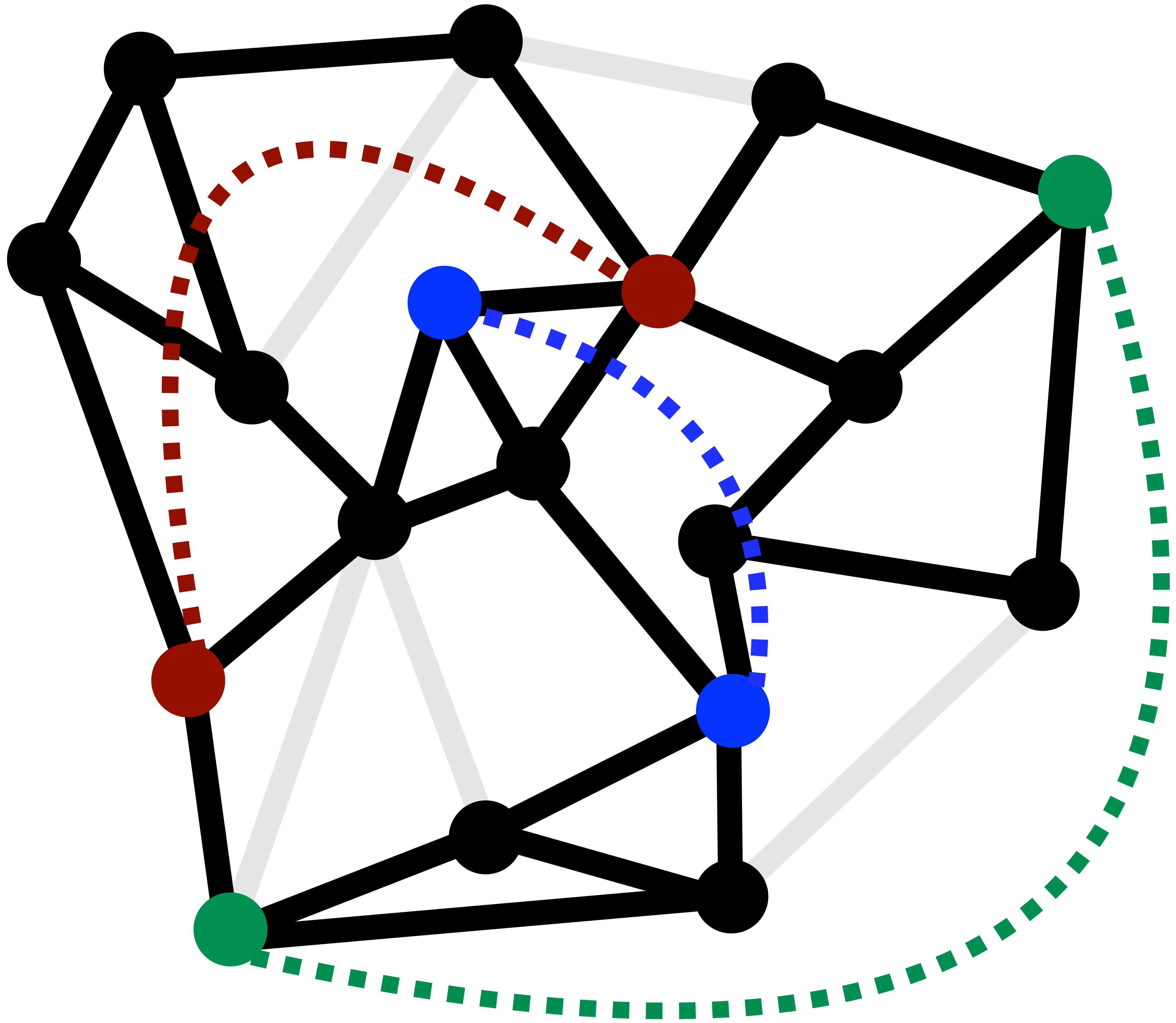
Background: Dynamic Algorithms



Problem: demands don't just change, graph does too

Papers Overview

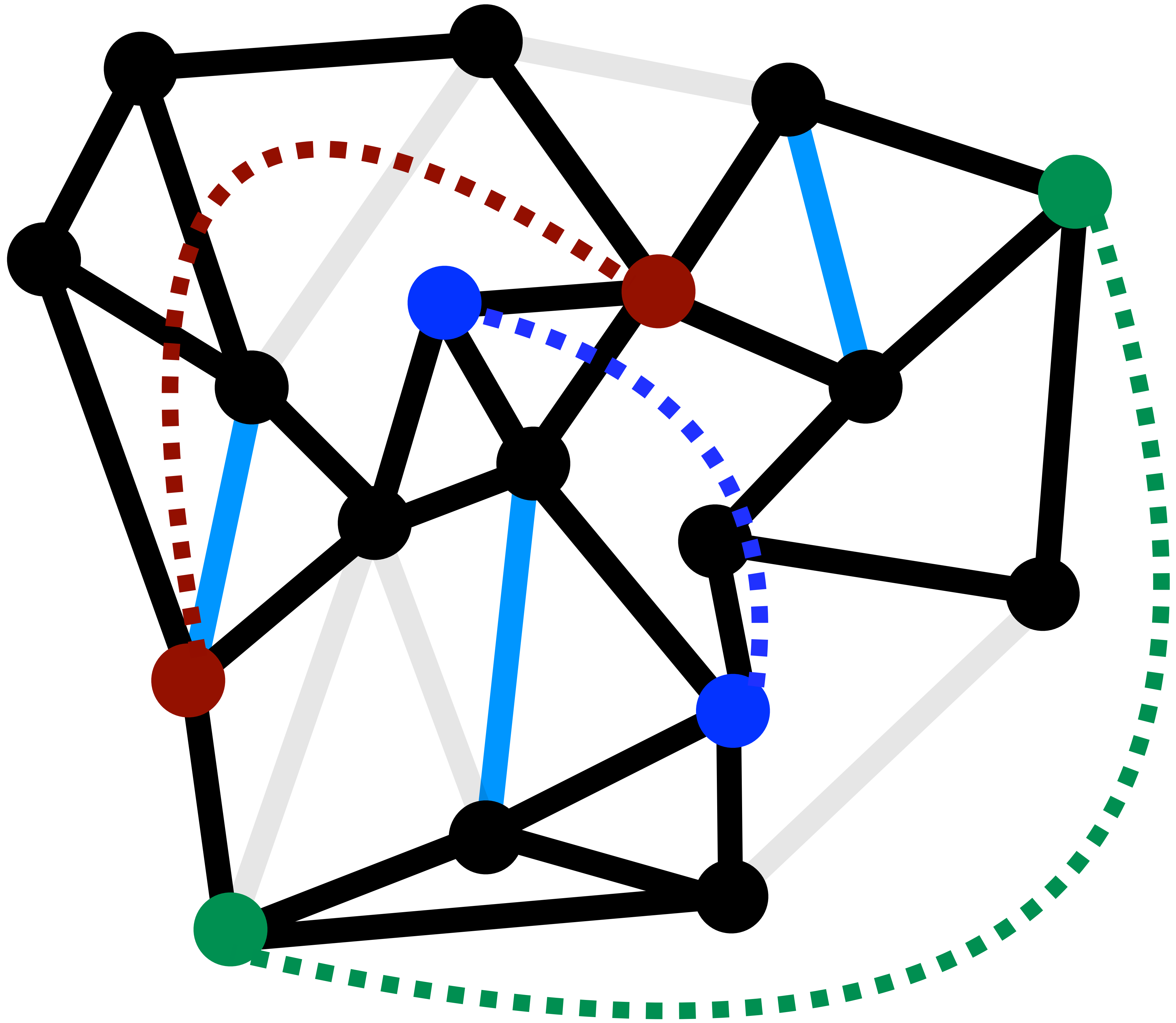
Background: Dynamic Algorithms



Problem: demands don't just change, graph does too

Papers Overview

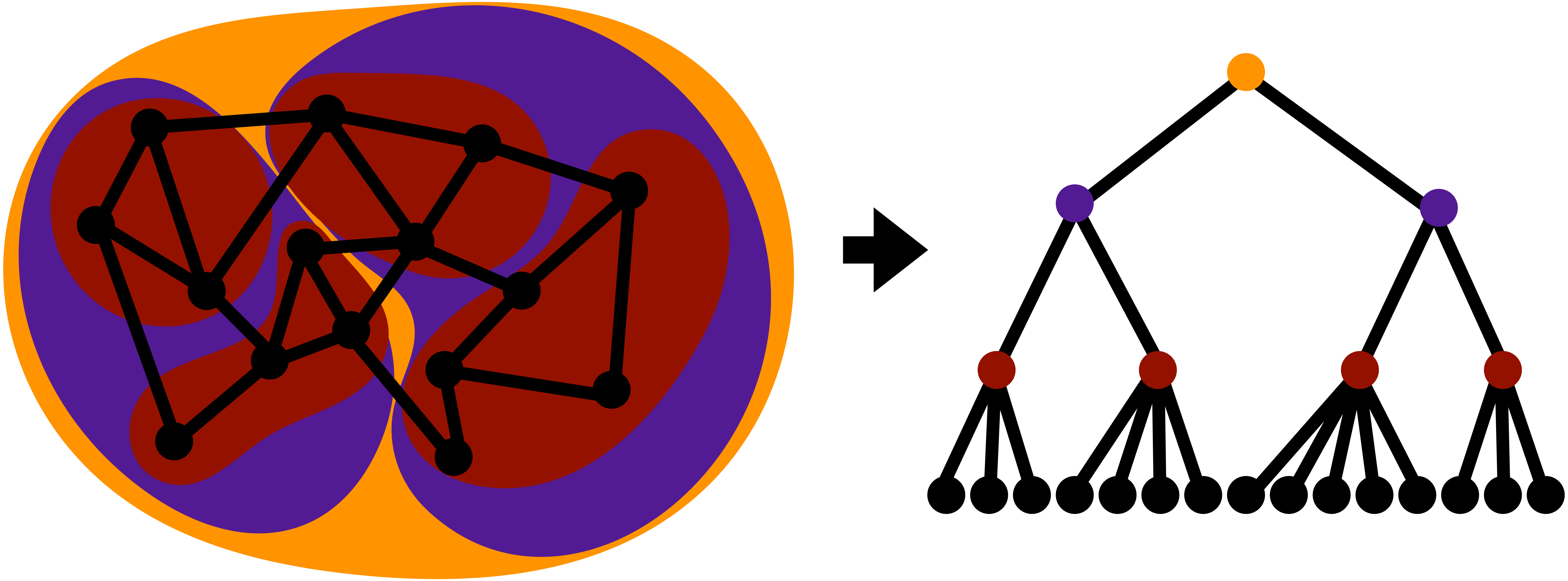
Background: Dynamic Algorithms



Problem: demands don't just change, graph does too

Papers Overview

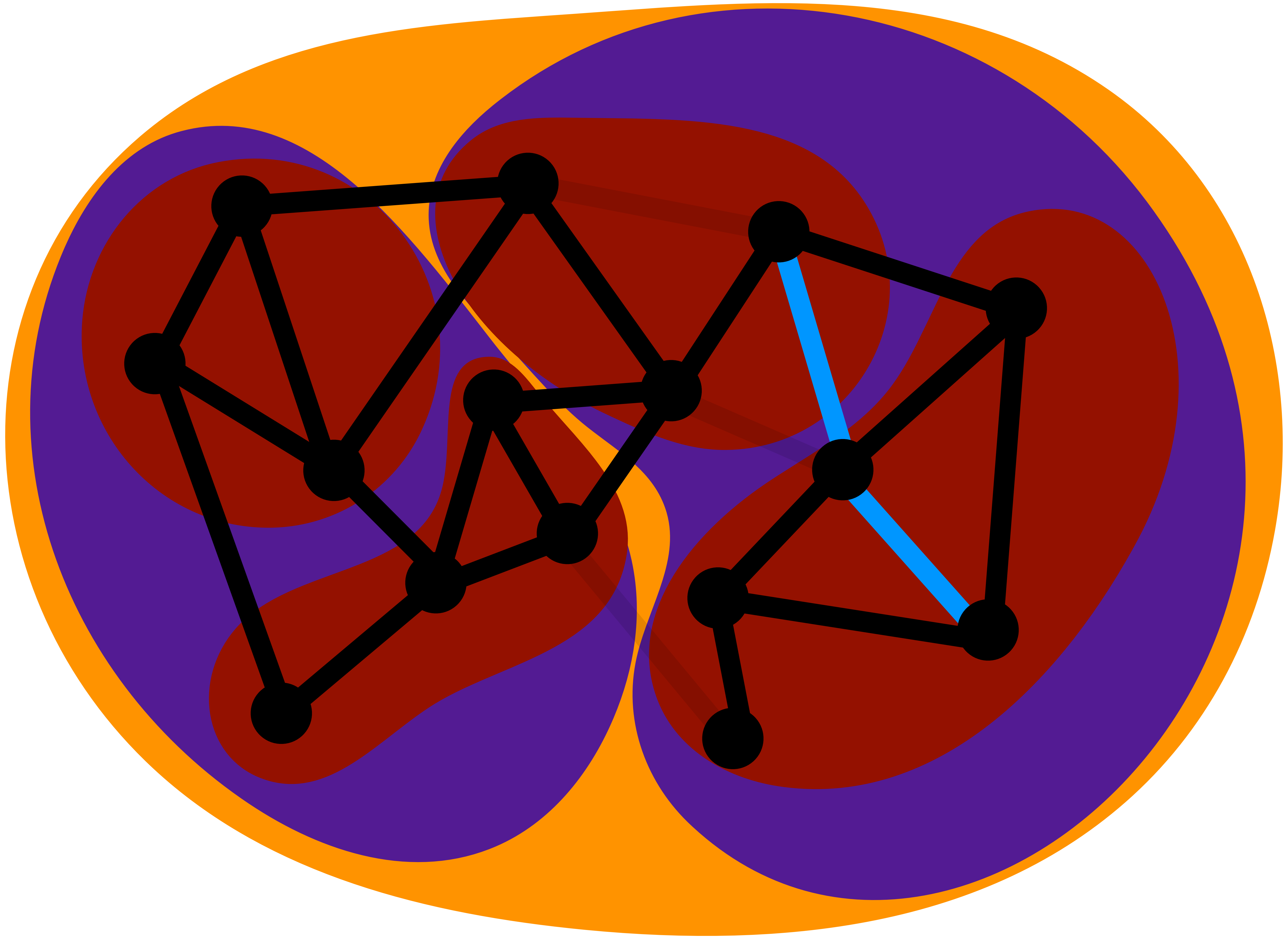
Background: Dynamic Algorithms

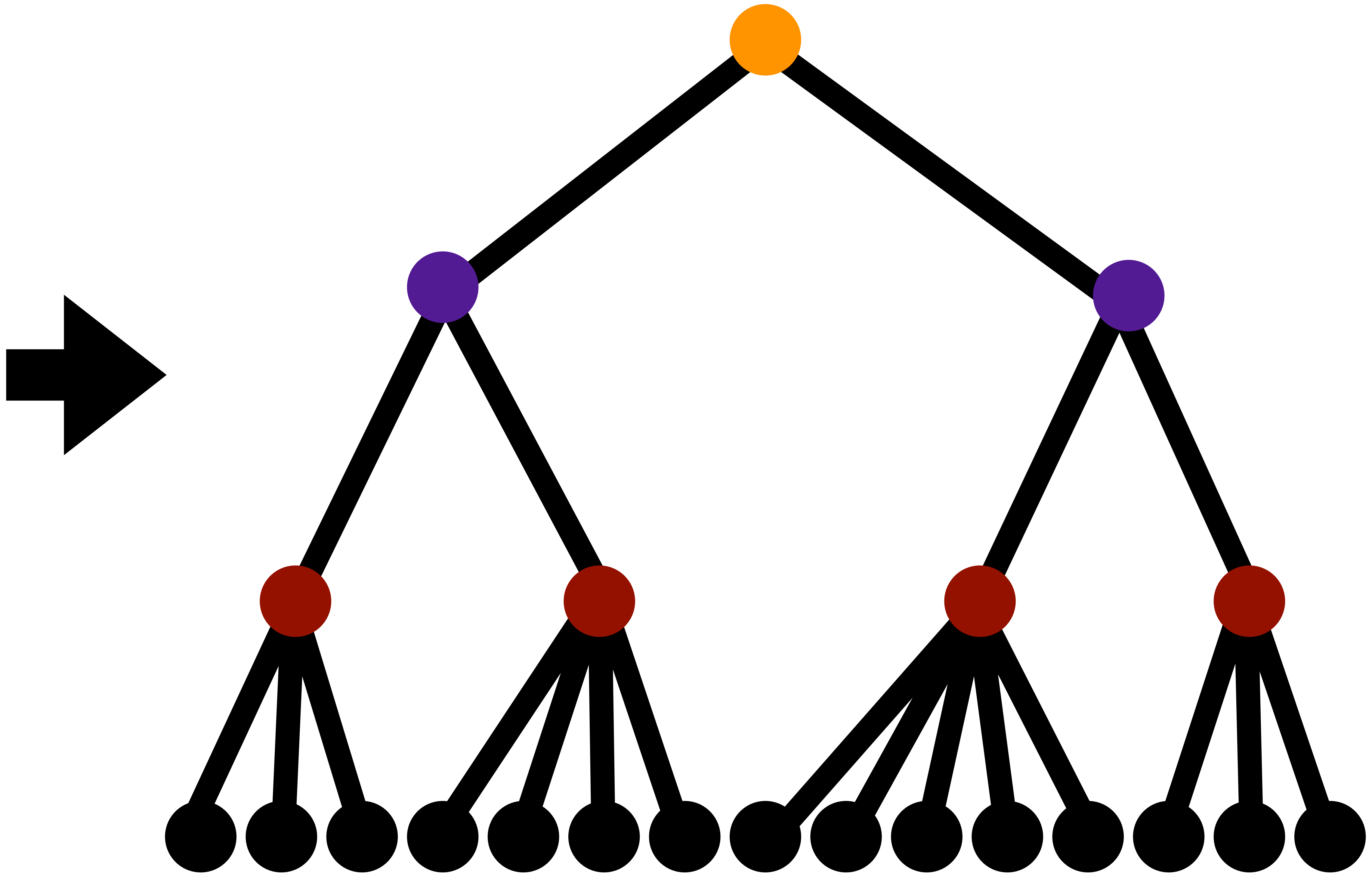


Problem: demands don't just change, graph does too

Papers Overview

Background: Dynamic Algorithms

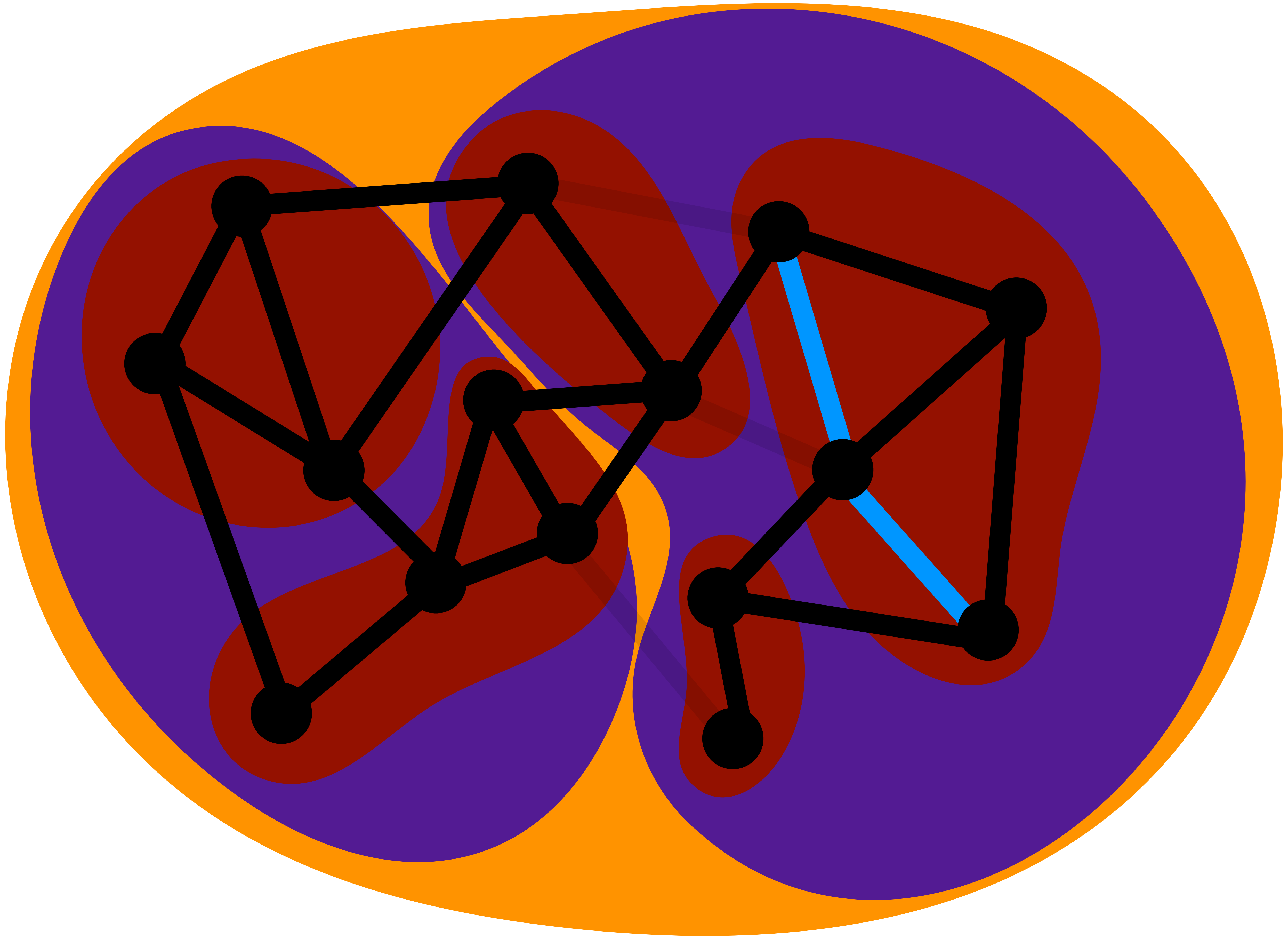


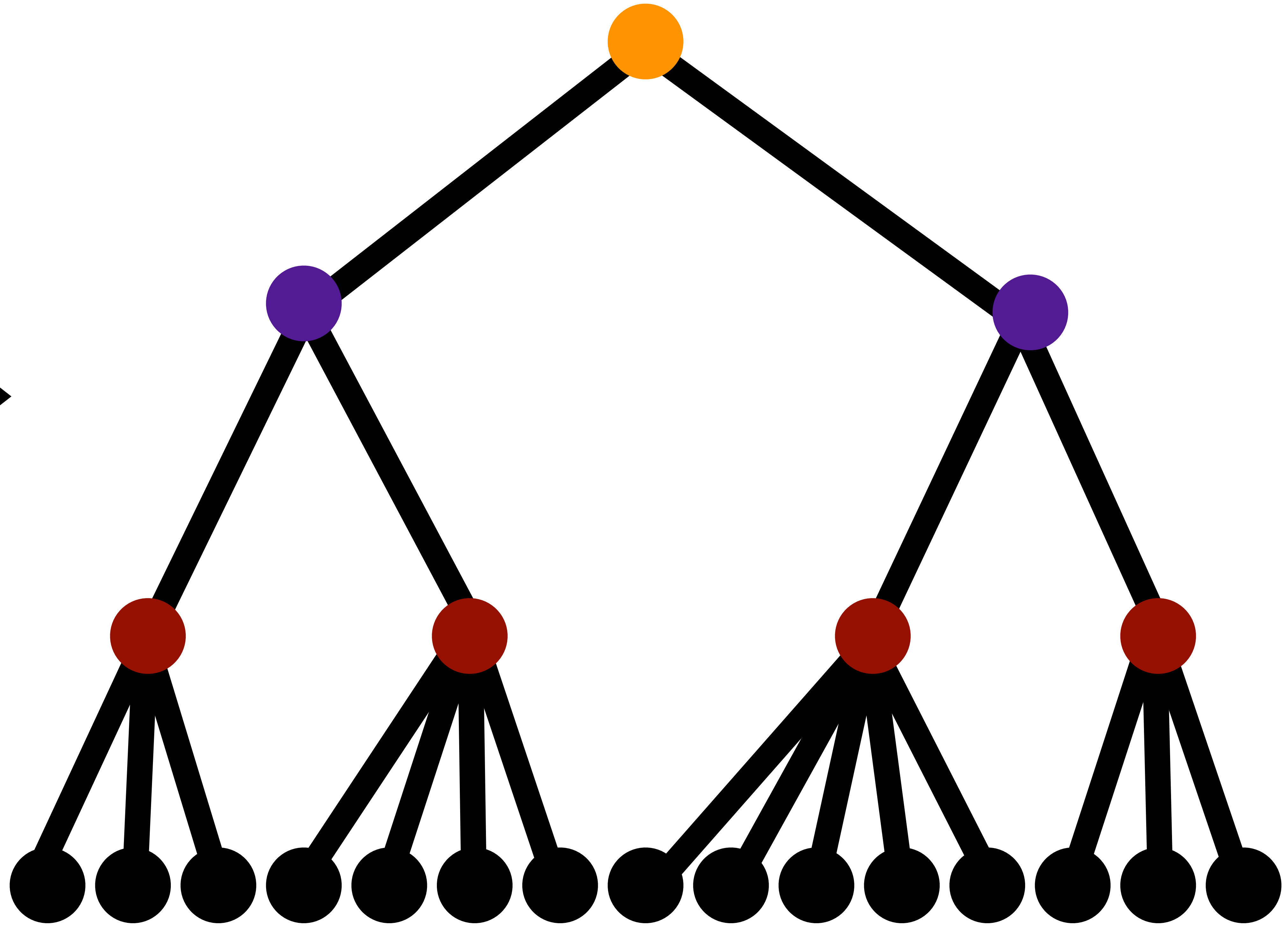


Problem: demands don't just change, graph does too

Papers Overview

Background: Dynamic Algorithms

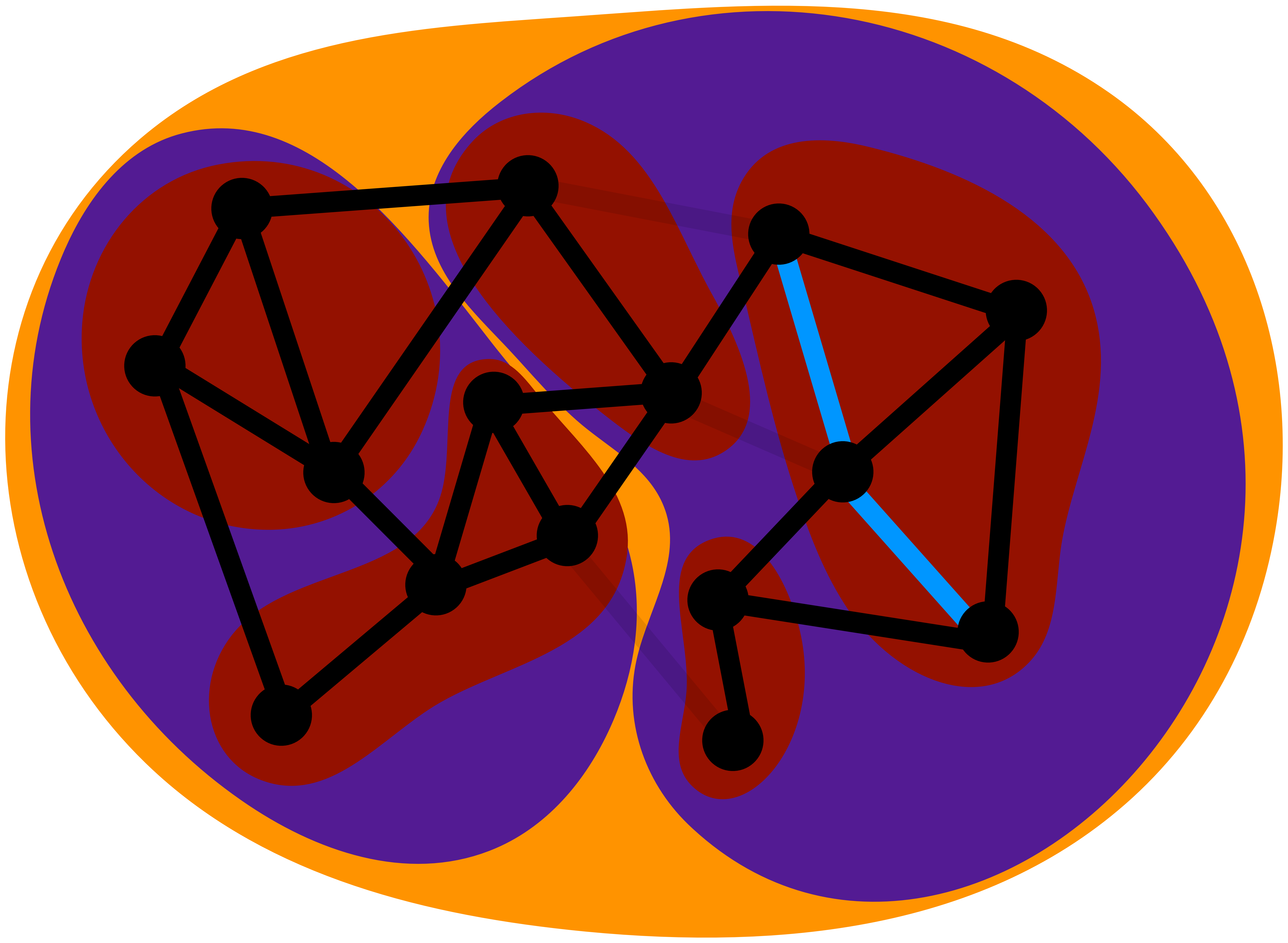




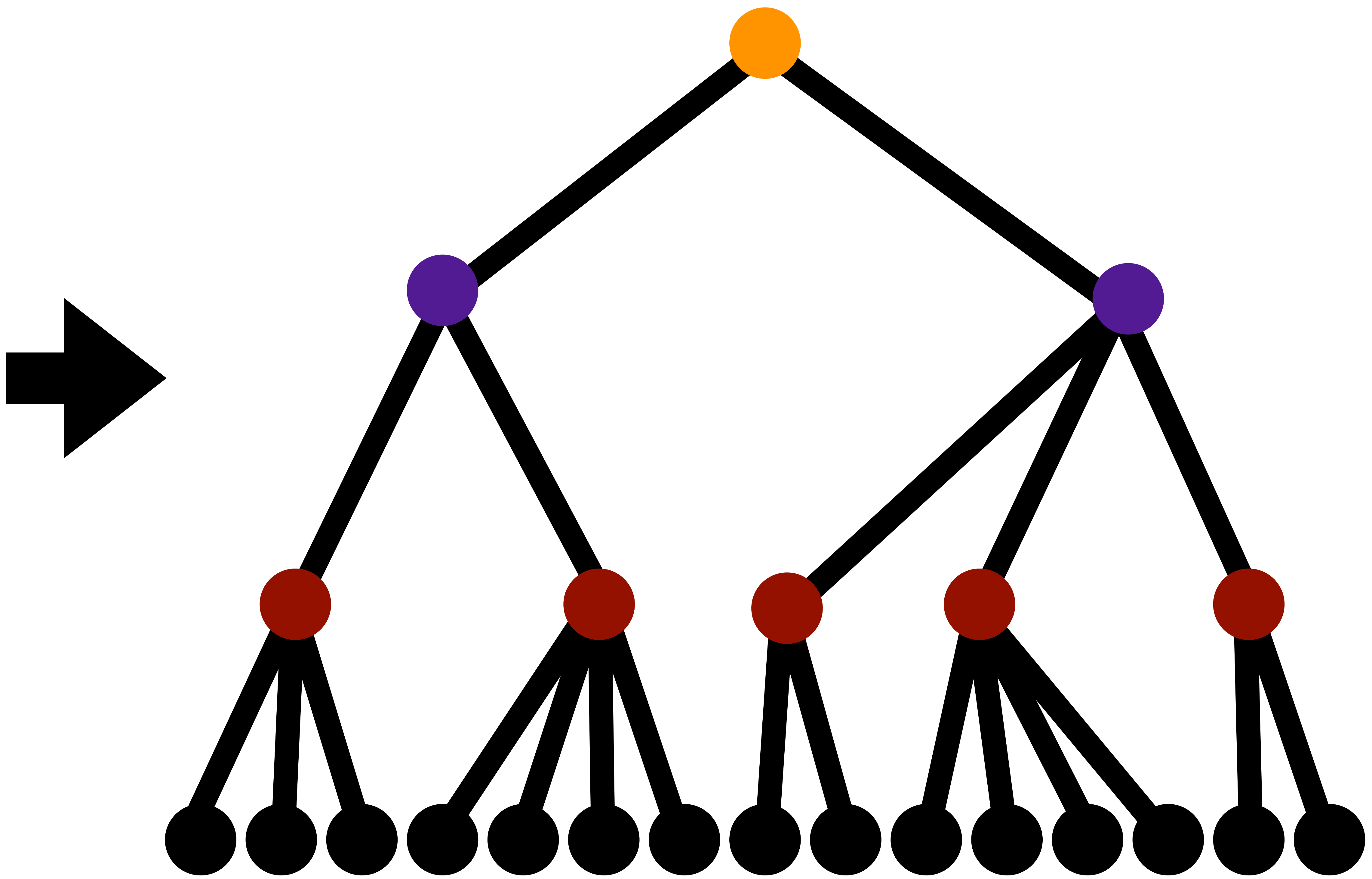
Problem: demands don't just change, graph does too

Papers Overview

Background: Dynamic Algorithms



Paper 6 brittle to changes 😭



Problem: demands don't just change, graph does too

Papers Overview

Background: Expander Graphs

- A "well-connected" graph
- **Conductance of cut** $S \subseteq V$ is

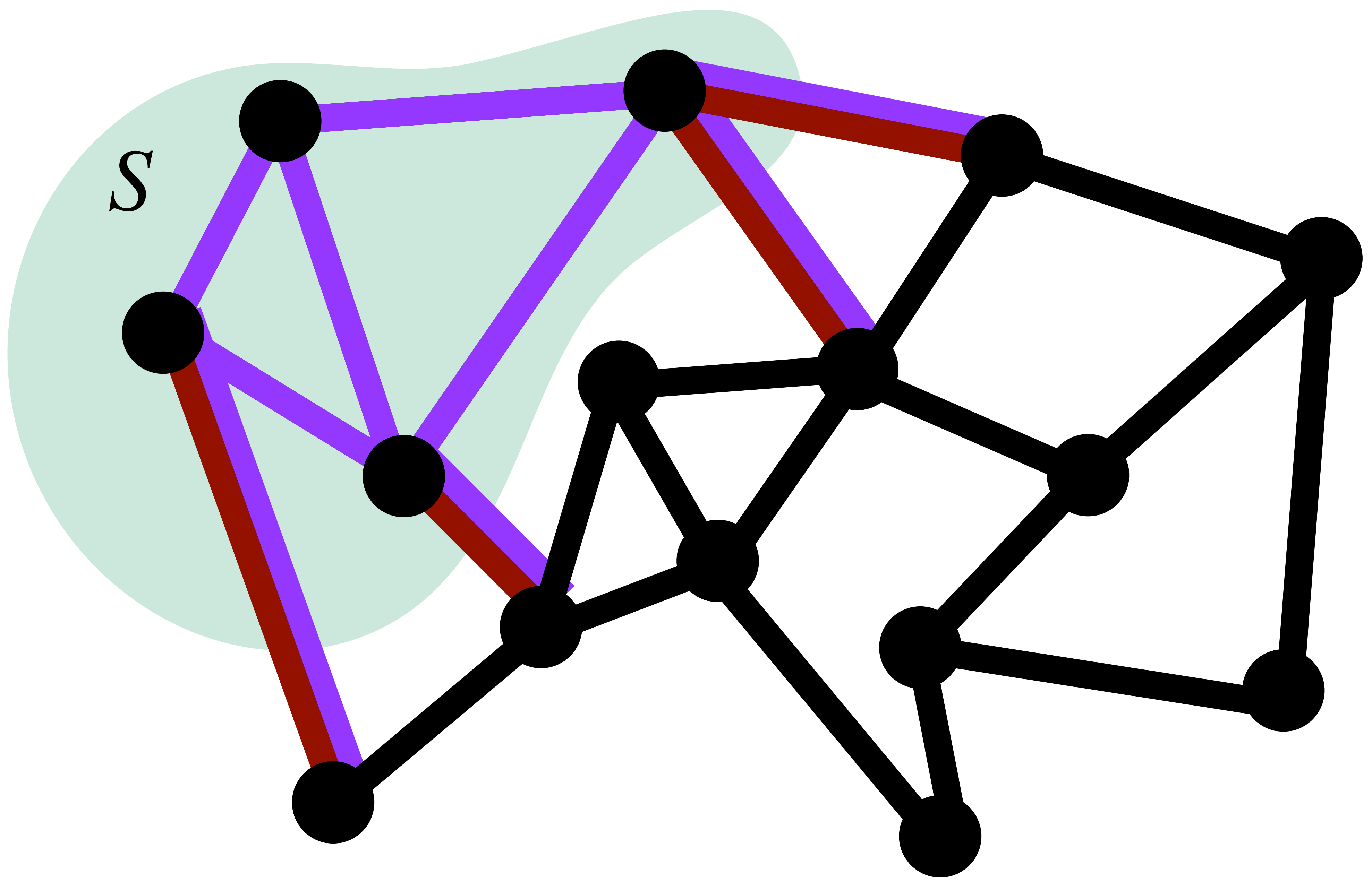
$$\phi(S) := |\delta(S)| \ / \ \mathrm{vol}(S)$$

where $\mathrm{vol}(S) := \sum_{v \in S} \deg(v)$

- **Conductance of graph** $G = (V, E)$ is

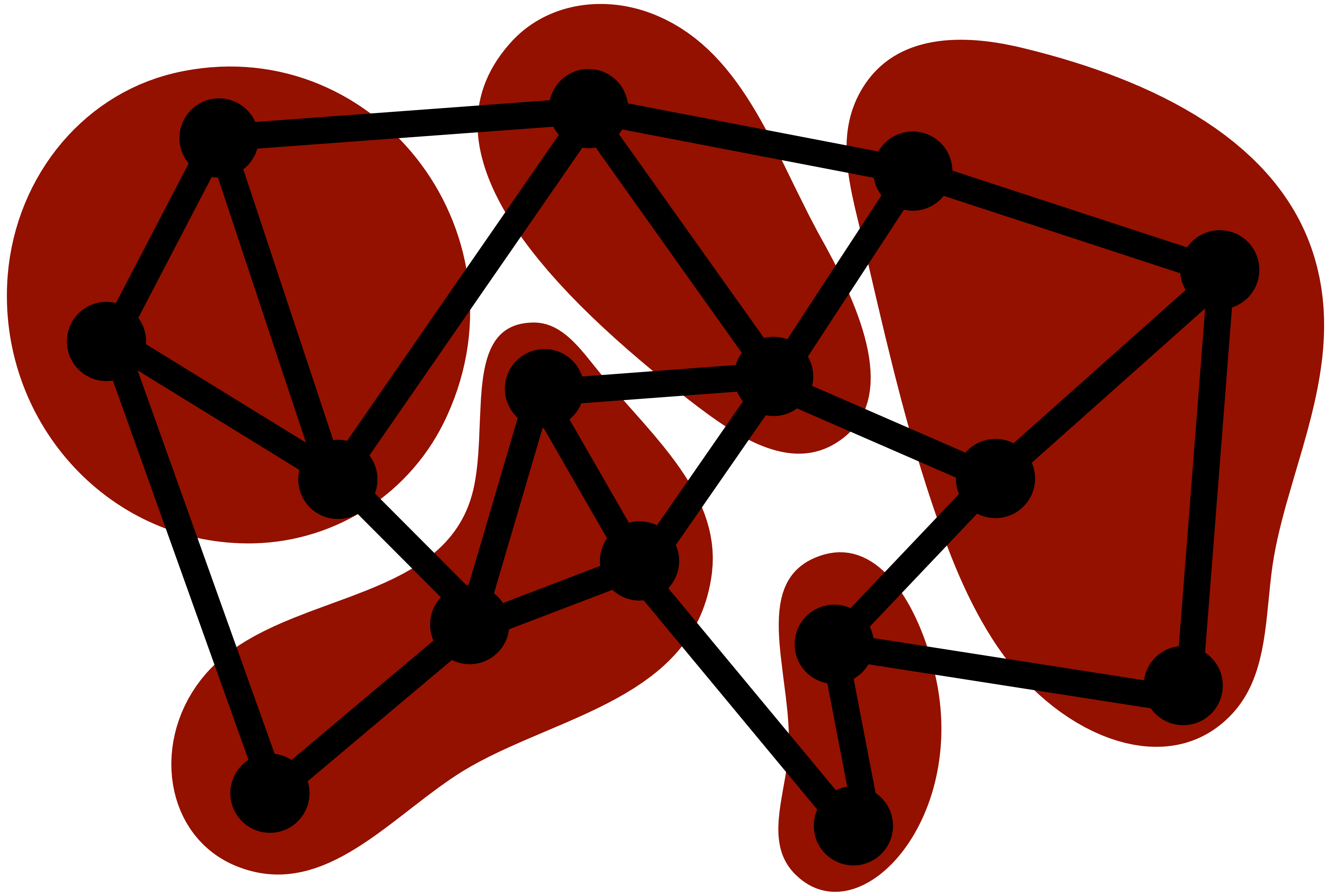
$$\phi_G := \min_{S \subseteq V} \phi(S)$$

- $G = (V, E)$ is a **ϕ-expander** if $\phi_G \geq \phi$



Papers Overview

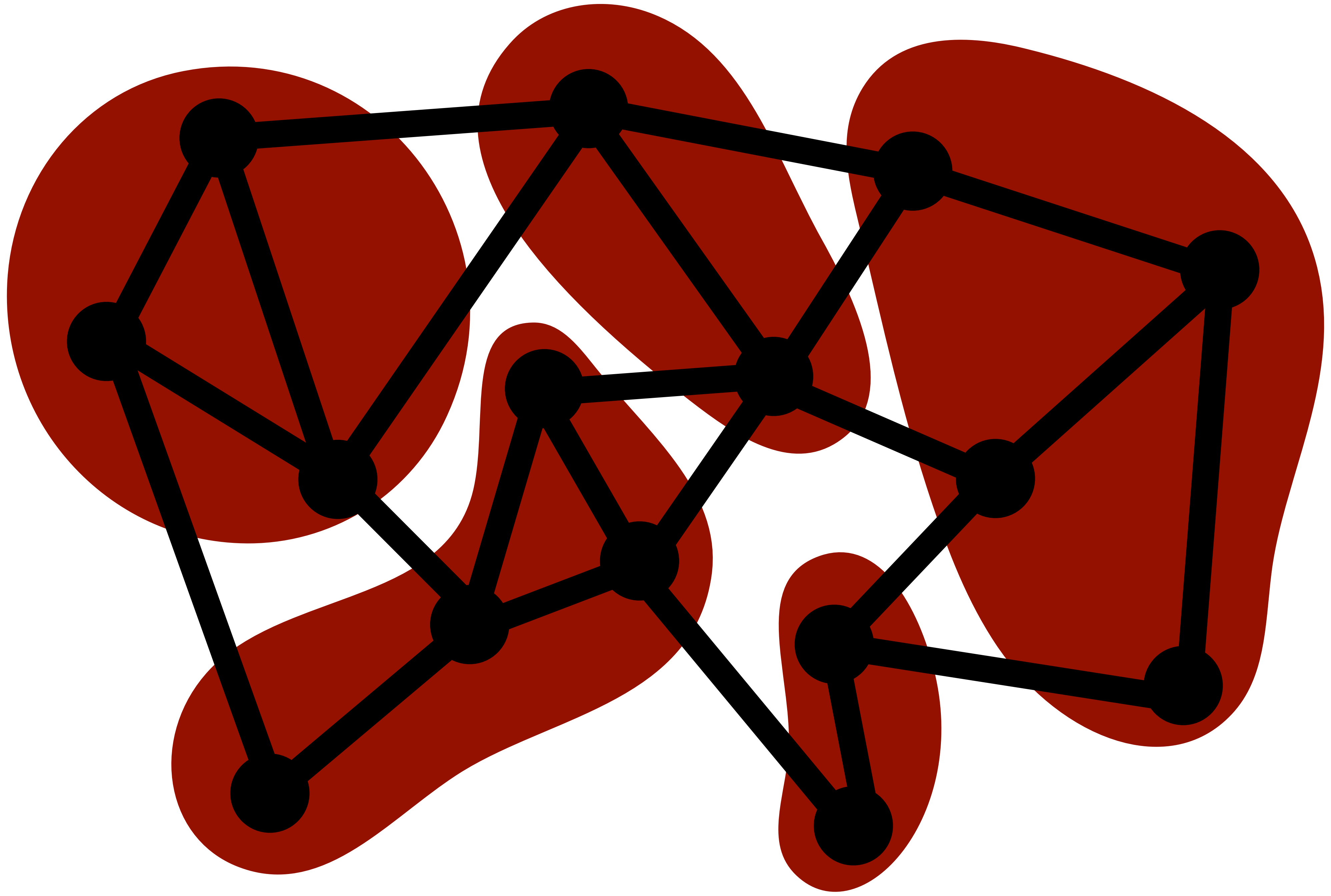
Paper 7: Expander Decompositions



Theorem(informal): "most" of a graph can be decomposed into expanders

Papers Overview

Paper 7: Expander Decompositions



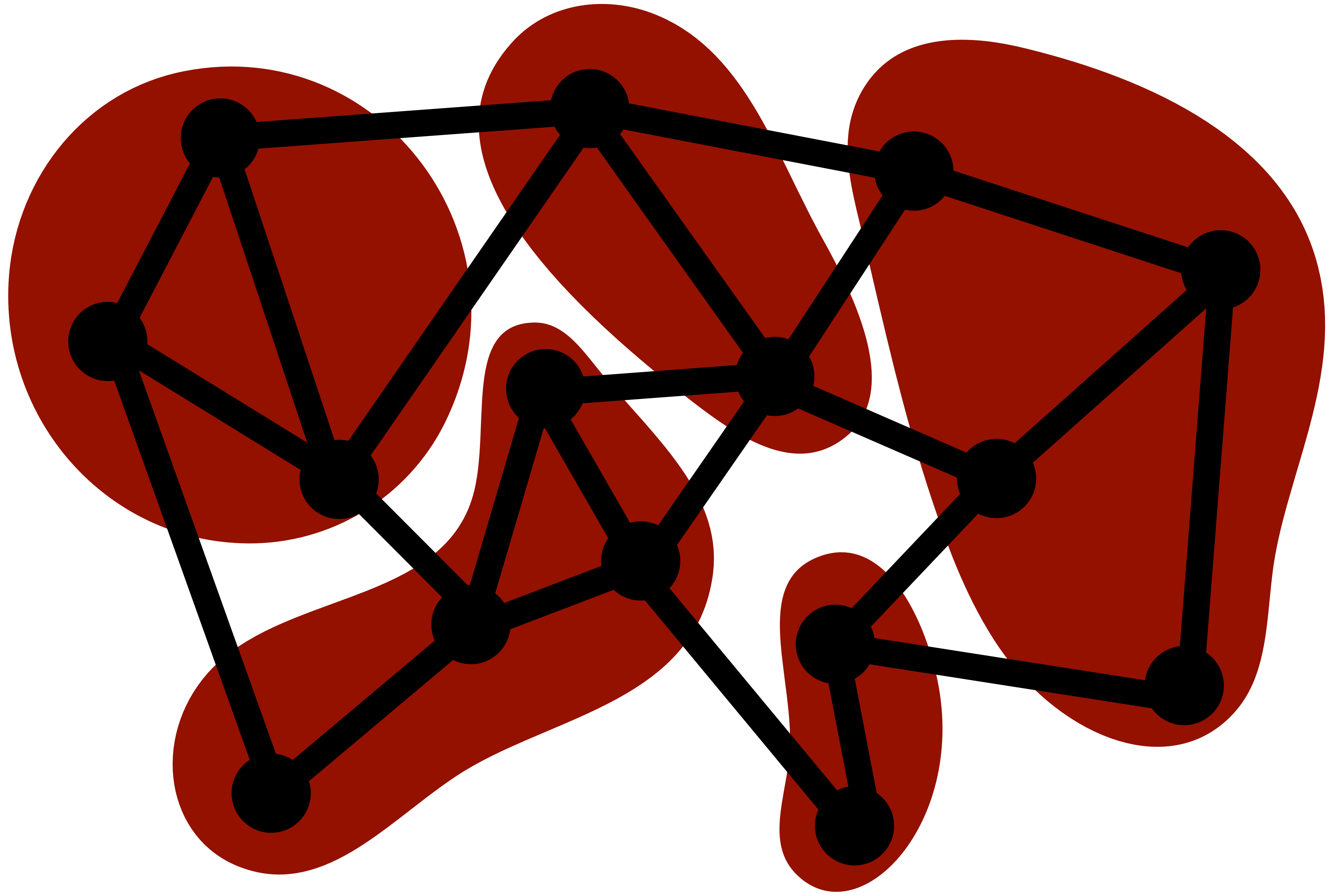
Very Hot Area of Algorithms

Theorem: vertices can be partitioned into $V_1, V_2, \ldots$ s.t.

1. $G[V_i]$ is a $\phi-$expander
2. at most $O(\phi \cdot \log n \cdot m)$ edges "cut"

Papers Overview

Paper 7: Expander Decompositions



Dynamic Tree Flow Sparsifiers?

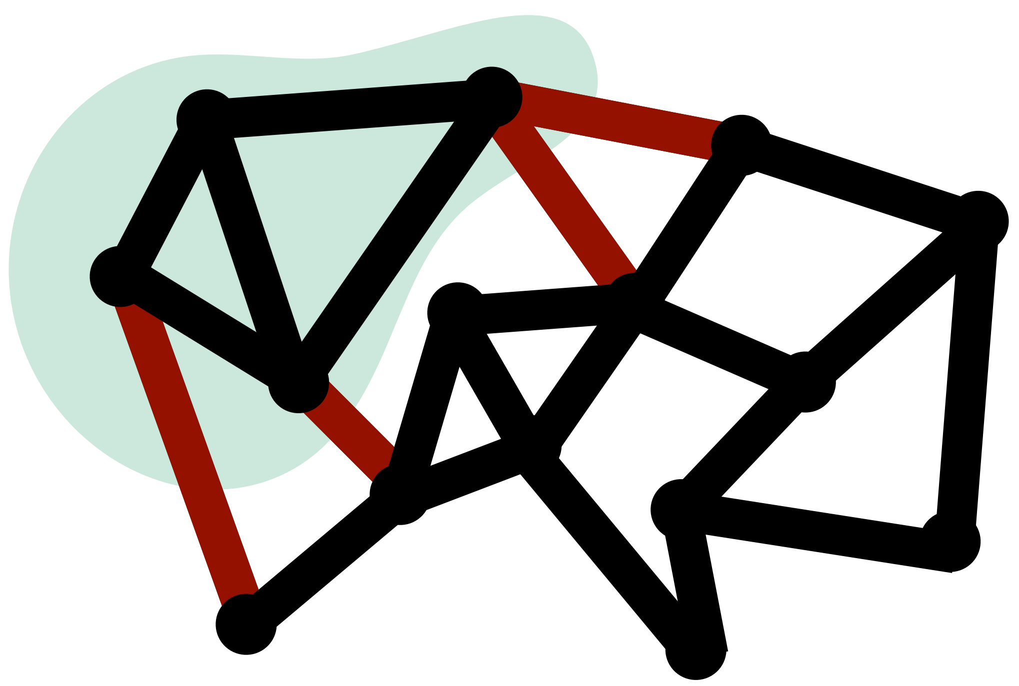
Theorem: vertices can be partitioned into $V_1, V_2, \ldots$ s.t.

1. $G[V_i]$ is a $\phi-$expander
2. at most $O(\phi \cdot \log n \cdot m)$ edges "cut"


Papers Overview
Flow / Cut Sparsification
graph $G = (V, E)$
4
7
5
Edge sparsification
graph $H = (V, E' \subseteq E)$
H cuts $\approx G$ cuts
(Random Sampling)
6
Structure sparsification
tree $T = (V, E')$
T flows $\approx G$ flows
(Tree Flow Sparsifiers)
8
Dynamic sparsification
tree $T = (V, E')$
T flows $\approx G$ flows
(Dynamic
Tree Flow Sparsifiers)

Papers Overview

Flow / Cut Sparsification



graph $G = (V, E)$

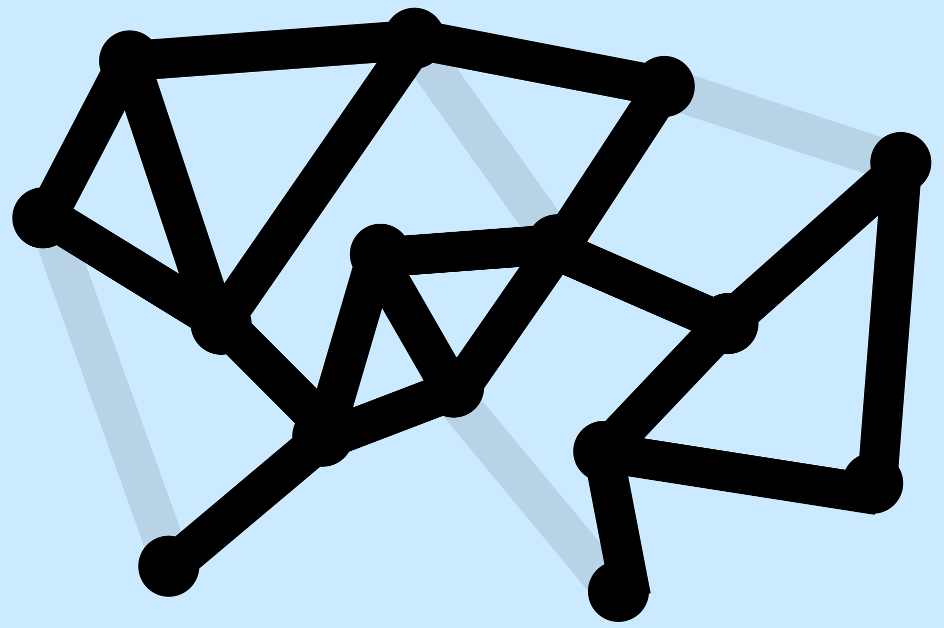
4 ✓ 5 ✓ 6 ✓ 7 ✓ 8

<u>Edge sparsification</u>

graph $H = (V, E' \subseteq E)$

H cuts $\approx G$ cuts

(Random Sampling)



<u>*Structure sparsification*</u>

tree $T = (V, E')$

T flows $\approx G$ flows

(Tree Flow Sparsifiers)

<u>*Dynamic sparsification*</u>

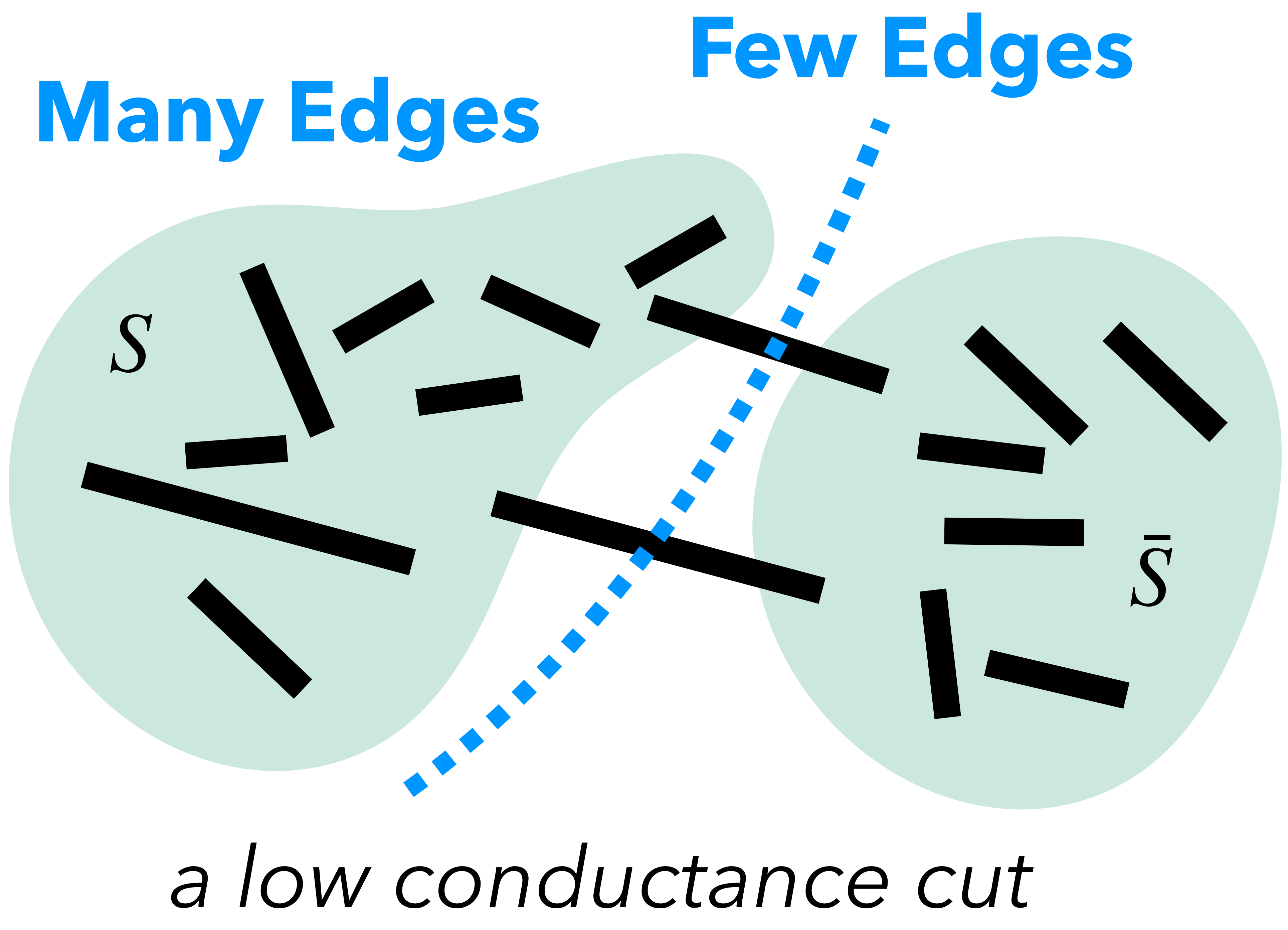
tree $T = (V, E')$

T flows $\approx G$ flows

(Dynamic Tree Flow Sparsifiers)

Papers Overview

Paper 8: Dynamic Tree Flow Sparsifiers (The Expander Hierarchy)

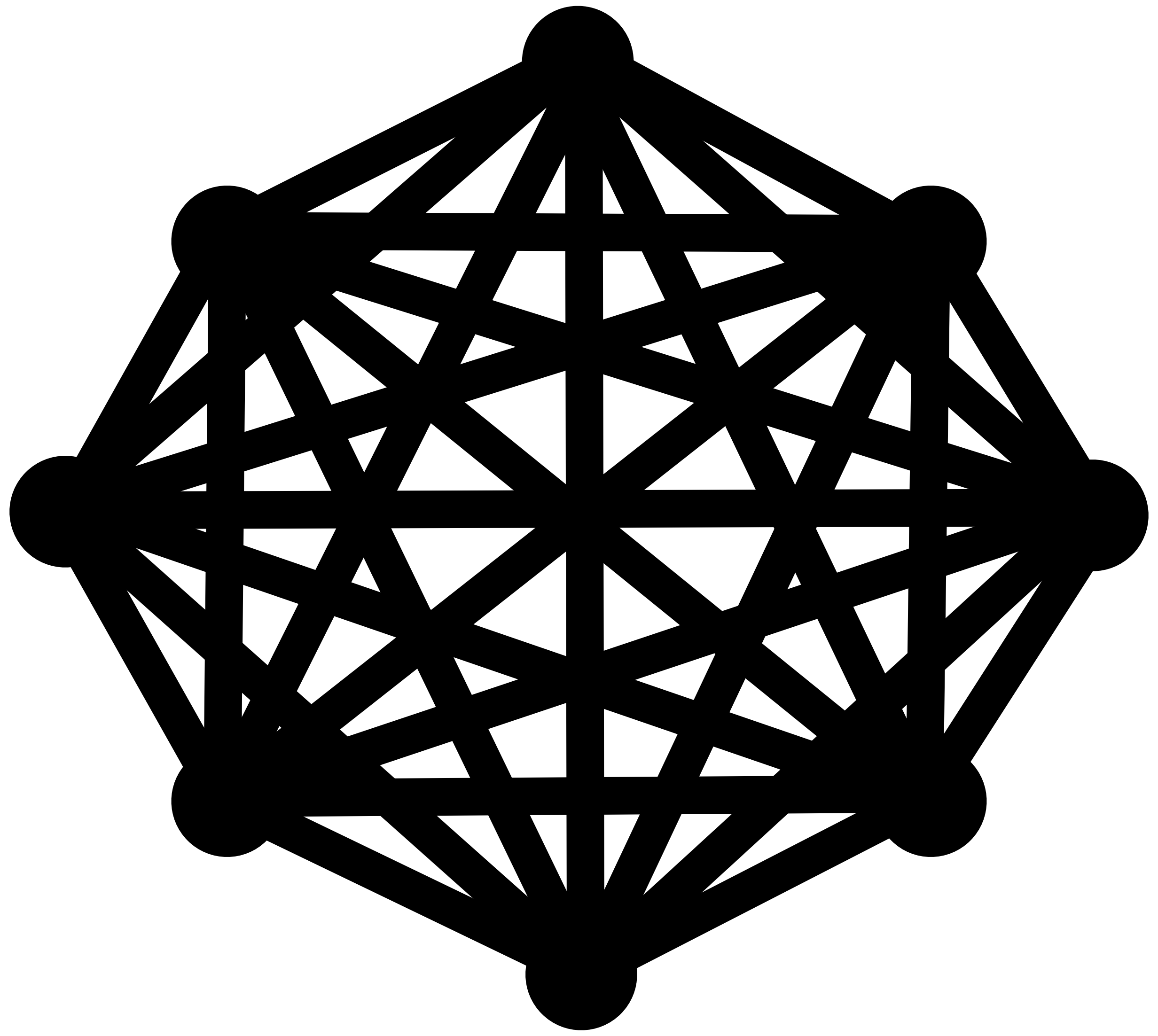


Expander
iff
low congestion flow

Intuition 1: expansion has *something* to do with flows

Papers Overview

Paper 8: Dynamic Tree Flow Sparsifiers (The Expander Hierarchy)

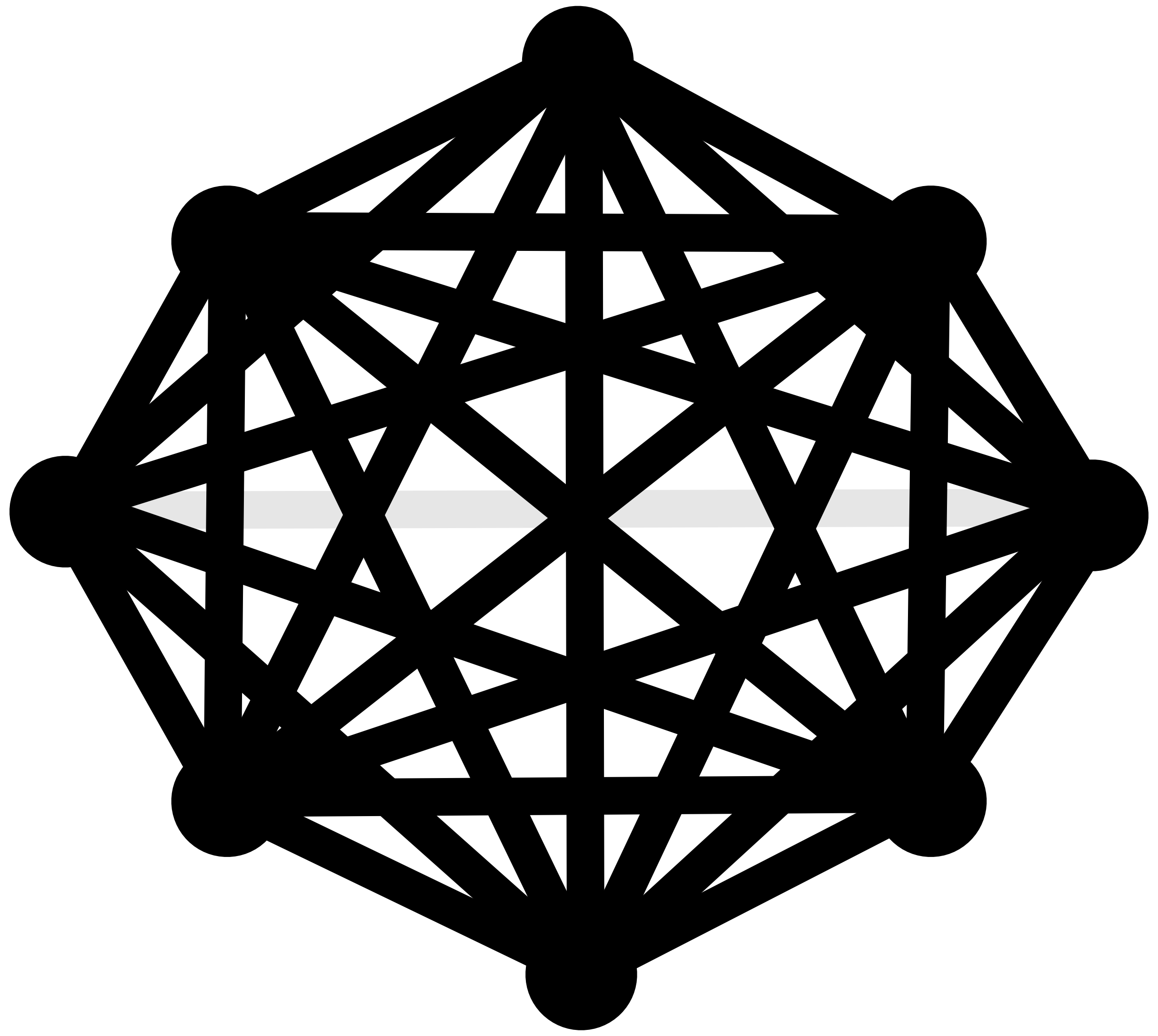


the most expandy expander

Intuition 2: expanders are *robust* to edge deletions

Papers Overview

Paper 8: Dynamic Tree Flow Sparsifiers (The Expander Hierarchy)

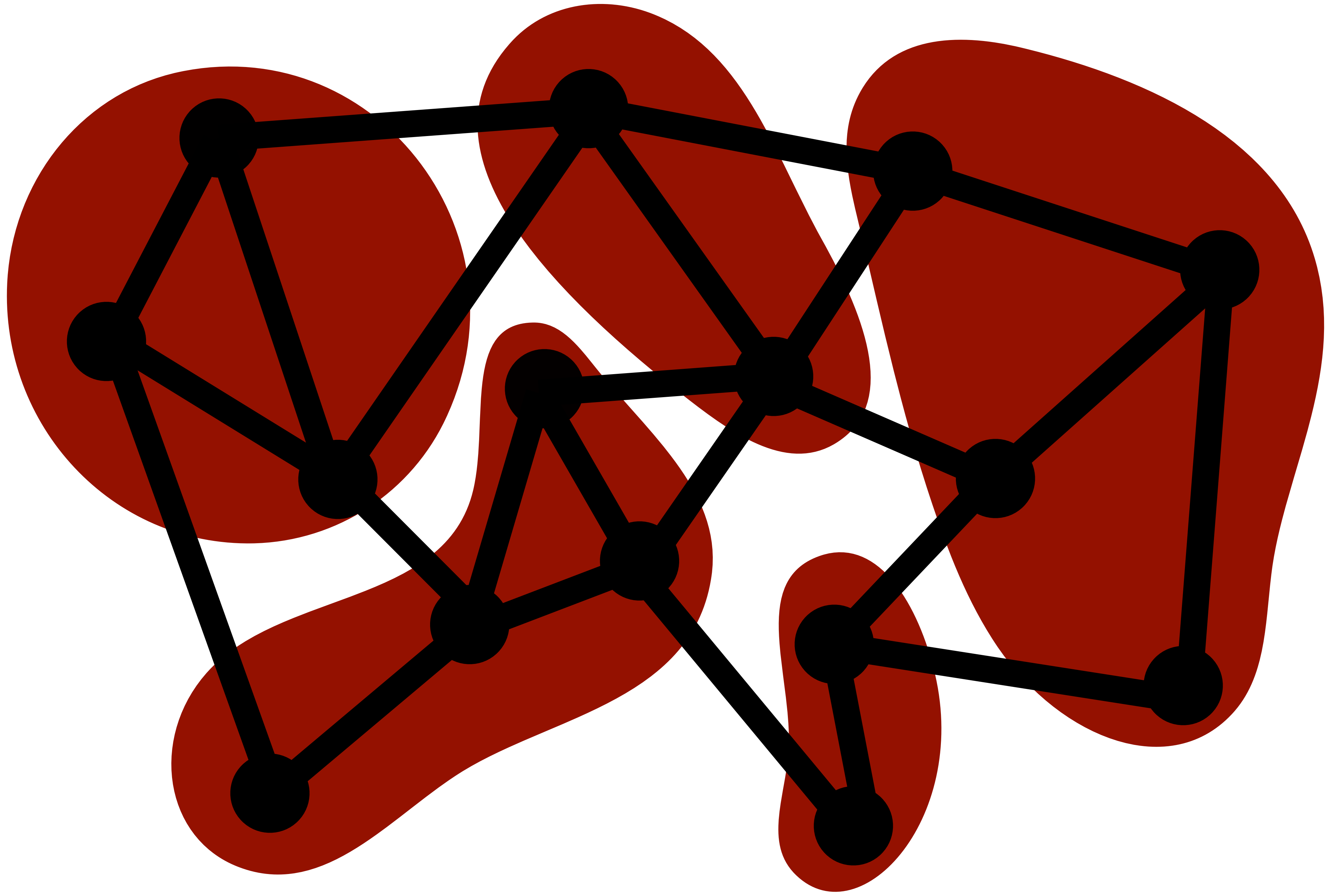


the most expandy expander

Intuition 2: expanders are *robust* to edge deletions

Papers Overview

Paper 8: Dynamic Tree Flow Sparsifiers (The Expander Hierarchy)



Theorem: vertices can be partitioned into $V_1, V_2, \ldots$ s.t.

1. $G[V_i]$ is a $\phi-$expander
2. at most $O(\phi \cdot \log n \cdot m)$ edges "cut"

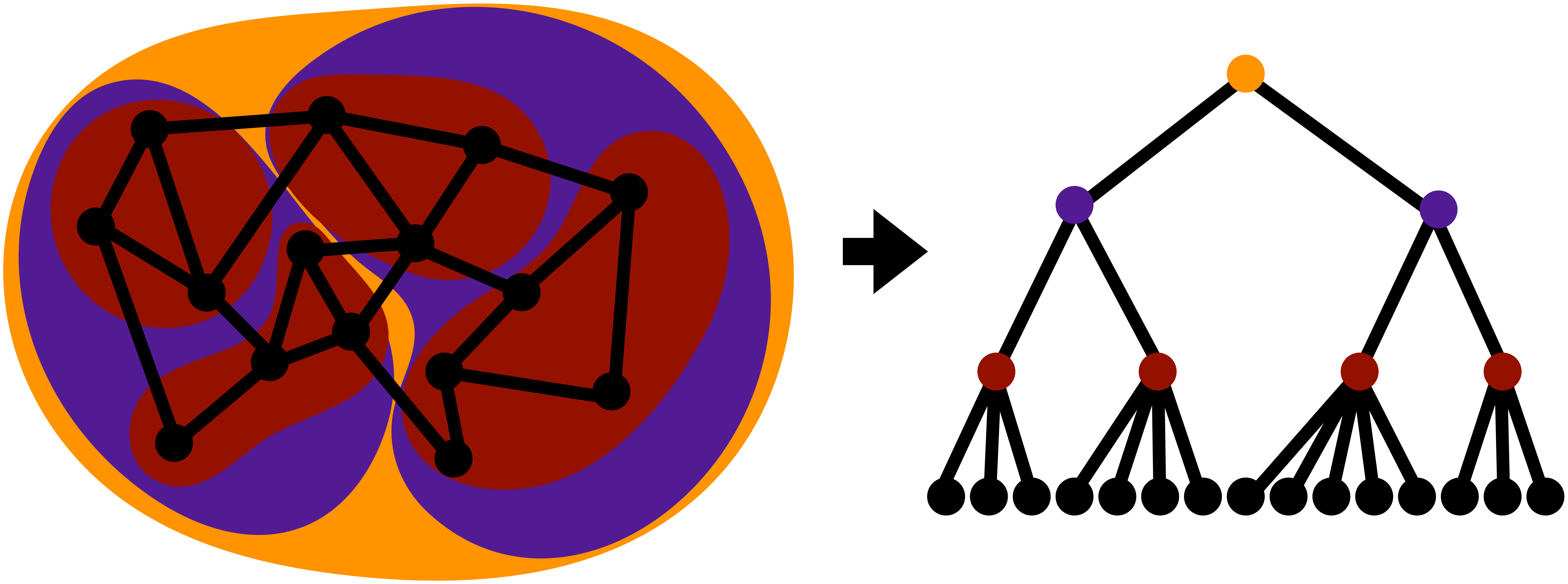
Papers Overview

Paper 8: Dynamic Tree Flow Sparsifiers (The Expander Hierarchy)

Theorem(informal): can construct a tree flow sparsifier robust to changes that is a hierarchy of expander decompositions by intuitions 1+2

Papers Overview

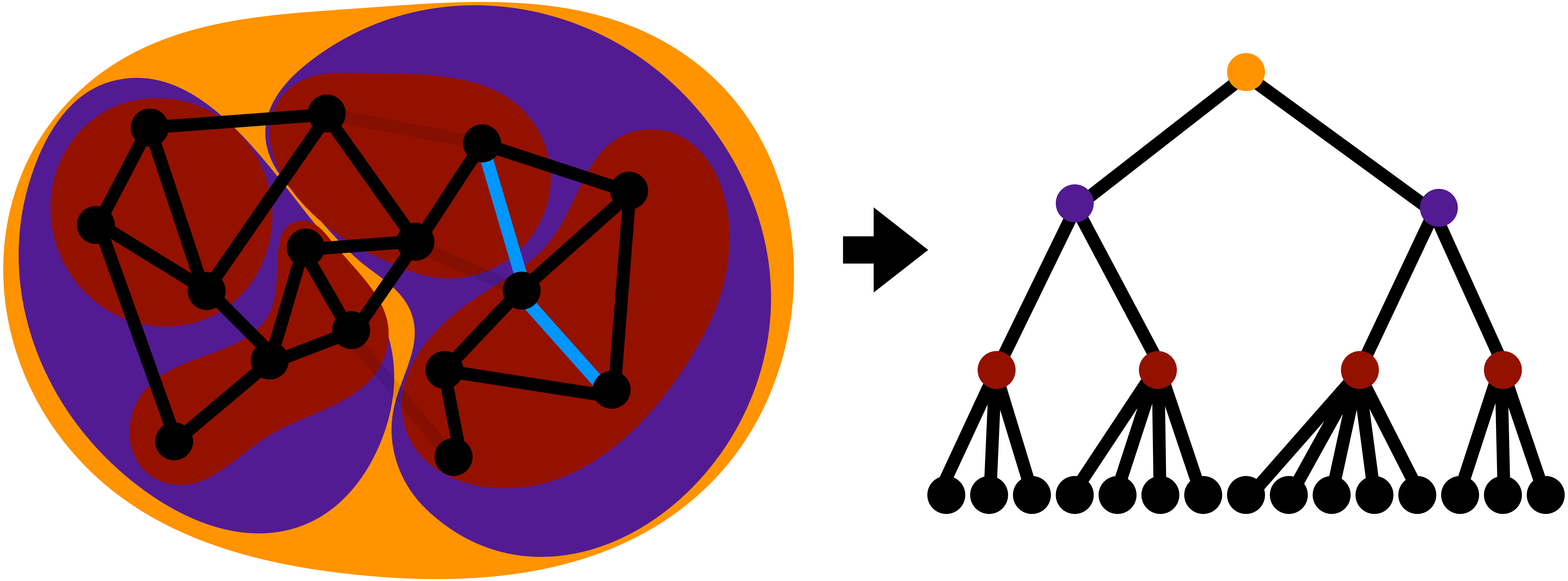
Paper 8: Dynamic Tree Flow Sparsifiers (The Expander Hierarchy)



Theorem(informal): can efficiently maintain a tree flow sparsifier under changes

Papers Overview

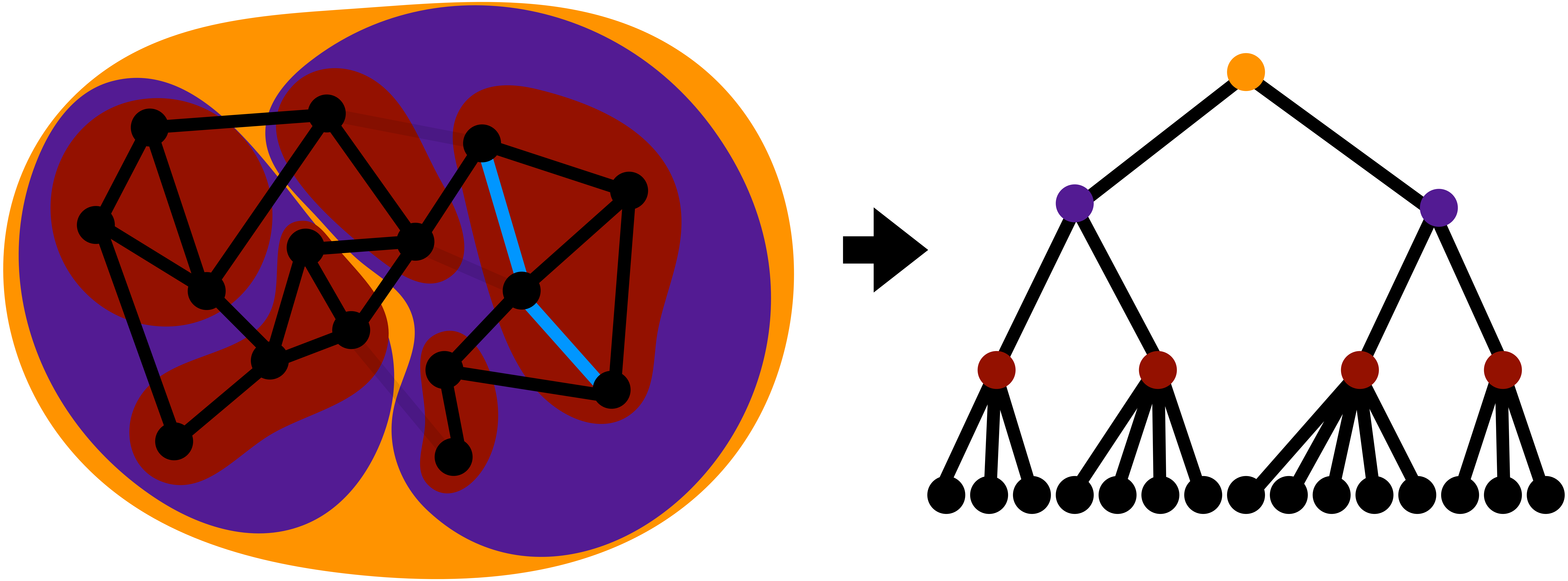
Paper 8: Dynamic Tree Flow Sparsifiers (The Expander Hierarchy)



Theorem(informal): can efficiently maintain a tree flow sparsifier under changes

Papers Overview

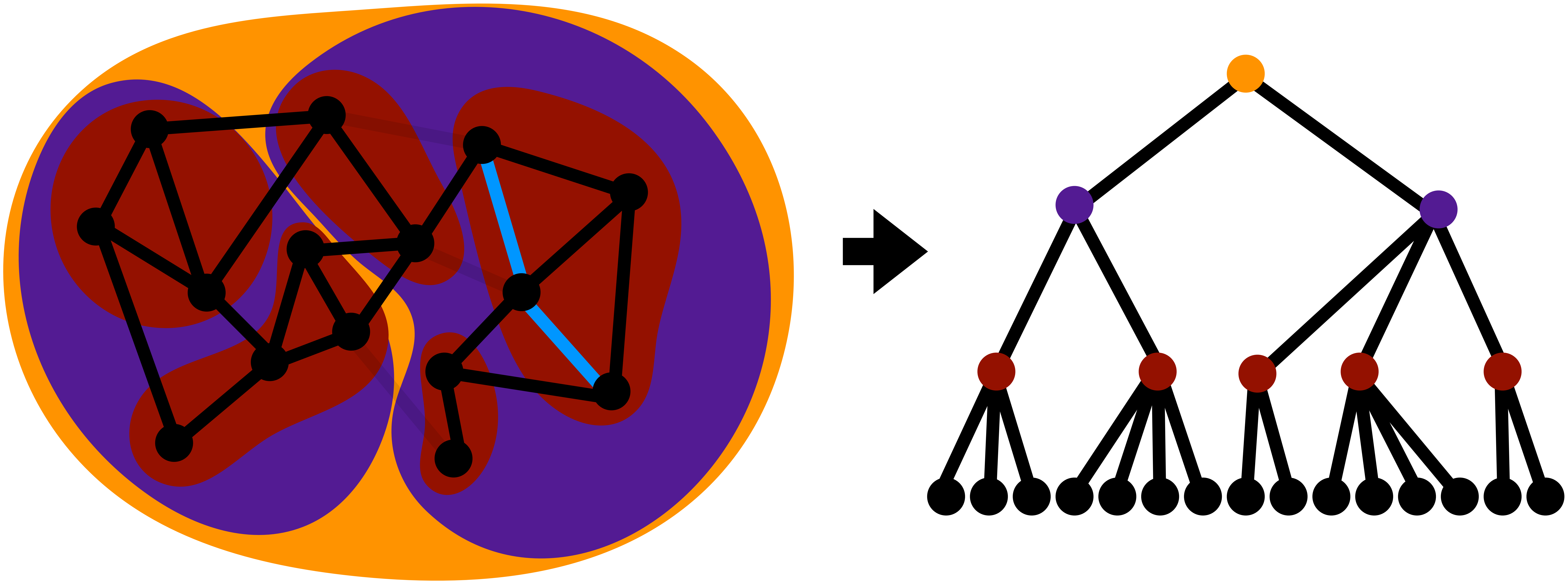
Paper 8: Dynamic Tree Flow Sparsifiers (The Expander Hierarchy)



Theorem(informal): can efficiently maintain a tree flow sparsifier under changes

Papers Overview

Paper 8: Dynamic Tree Flow Sparsifiers (The Expander Hierarchy)

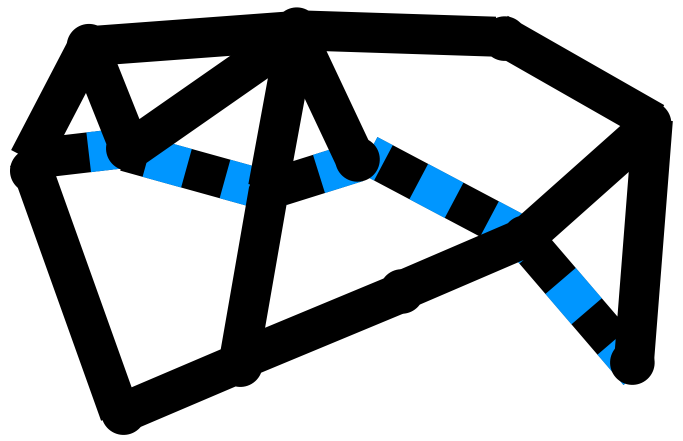


Theorem(informal): can efficiently maintain a tree flow sparsifier under changes

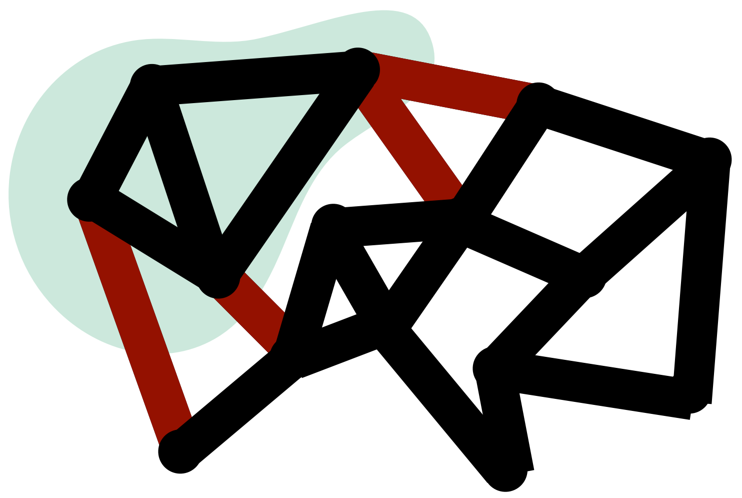
Papers Overview

Sparsification of Five Graph-Theoretic Objects

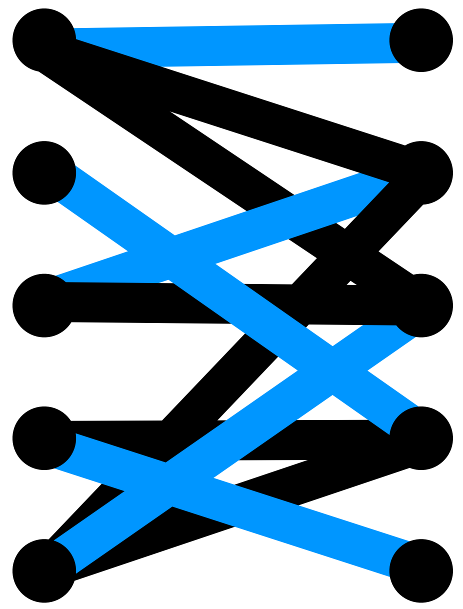
Distances



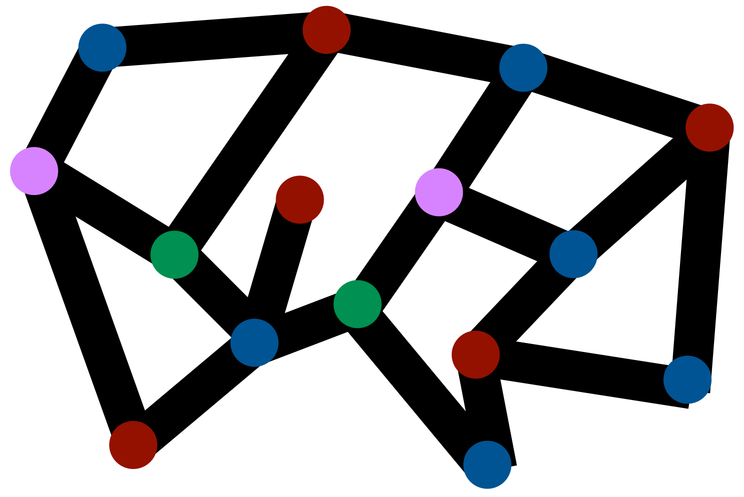
Cuts/Flows



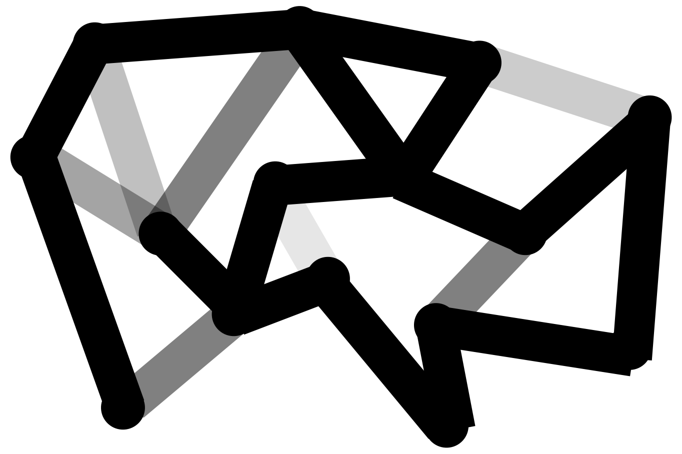
Matchings



Colorings



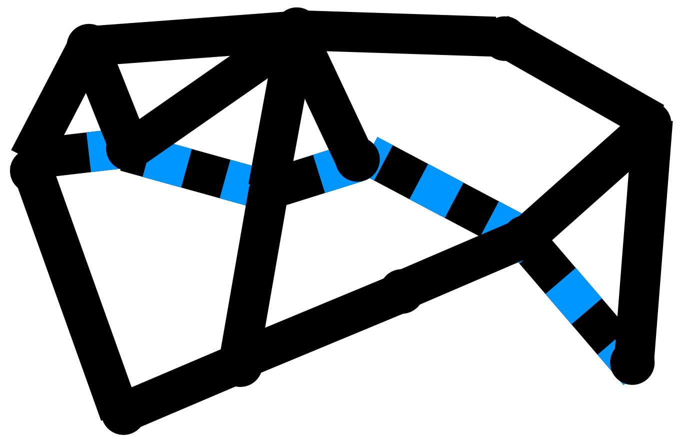
Fractional Opts



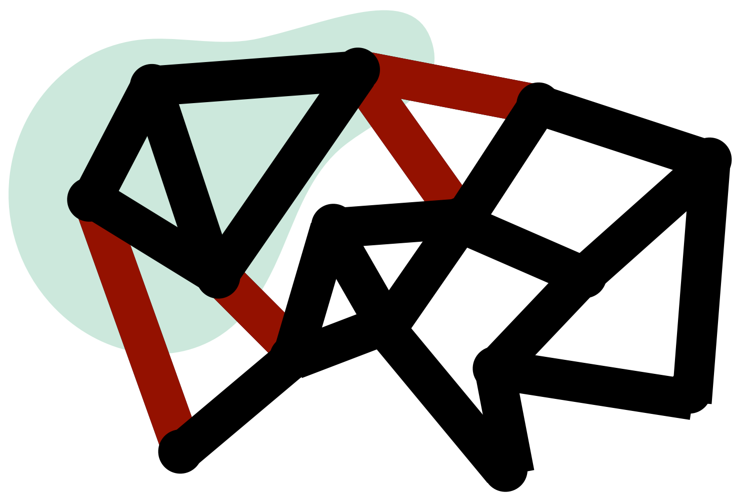
Papers Overview

Sparsification of Five Graph-Theoretic Objects

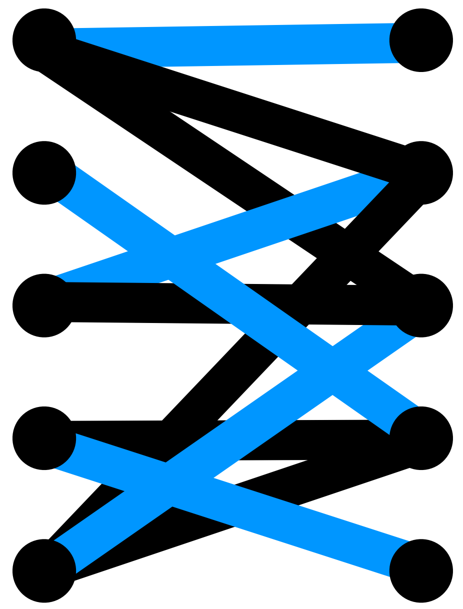
Distances



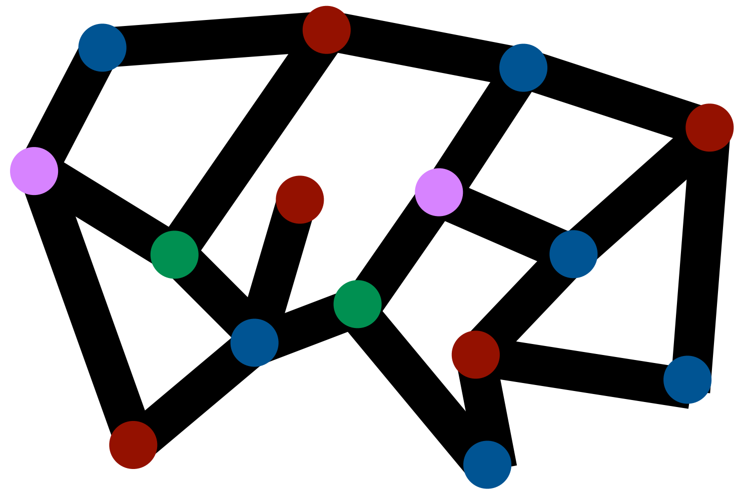
Cuts/Flows



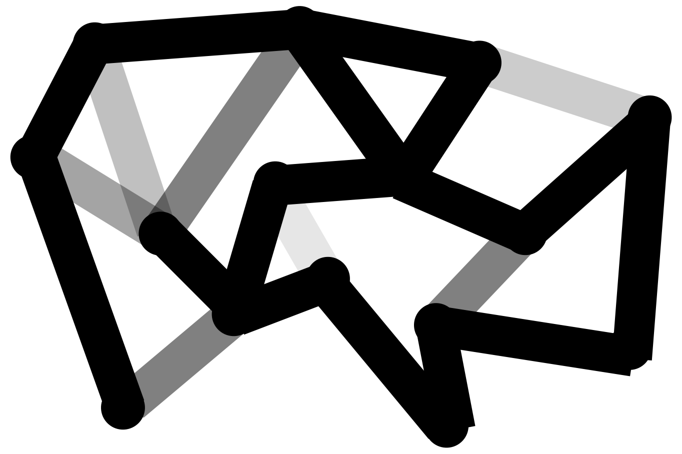
Matchings



Colorings

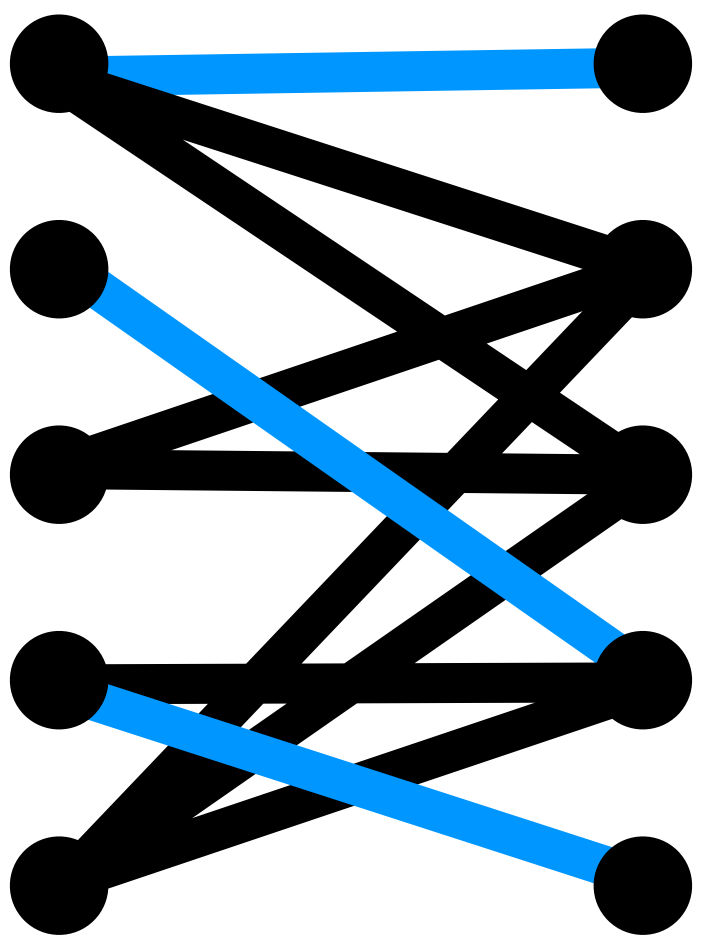


Fractional Opts

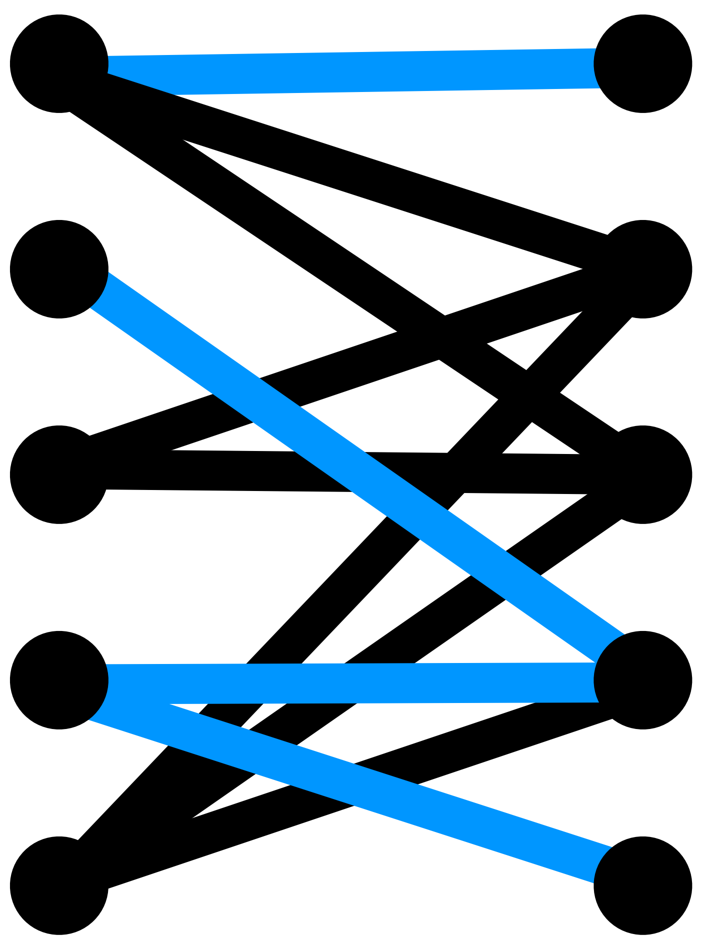


Papers Overview

Background: Matching Theory



matching

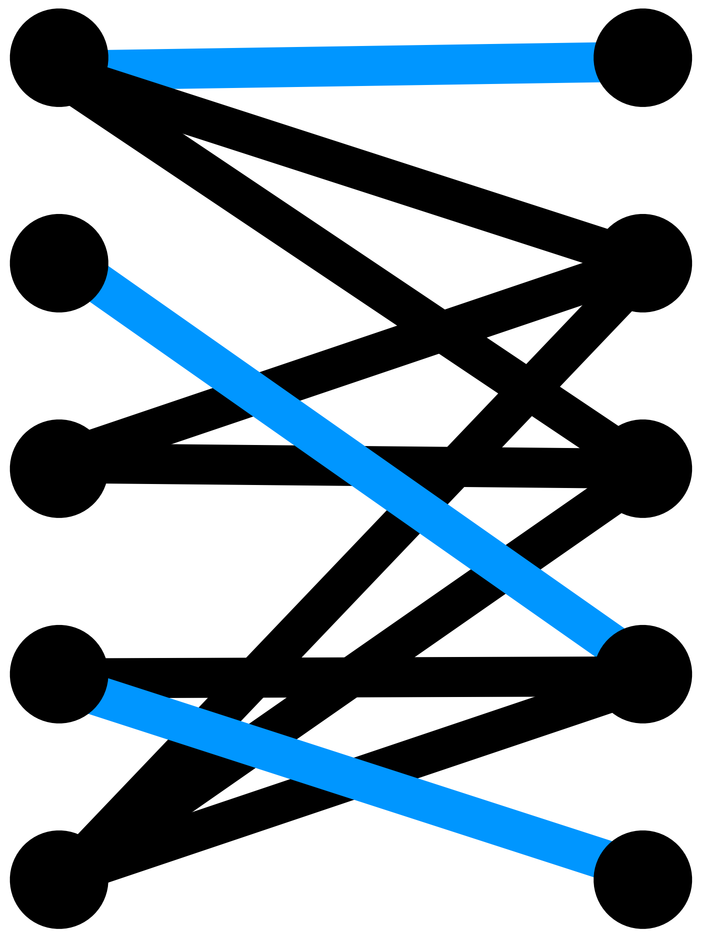


not a matching

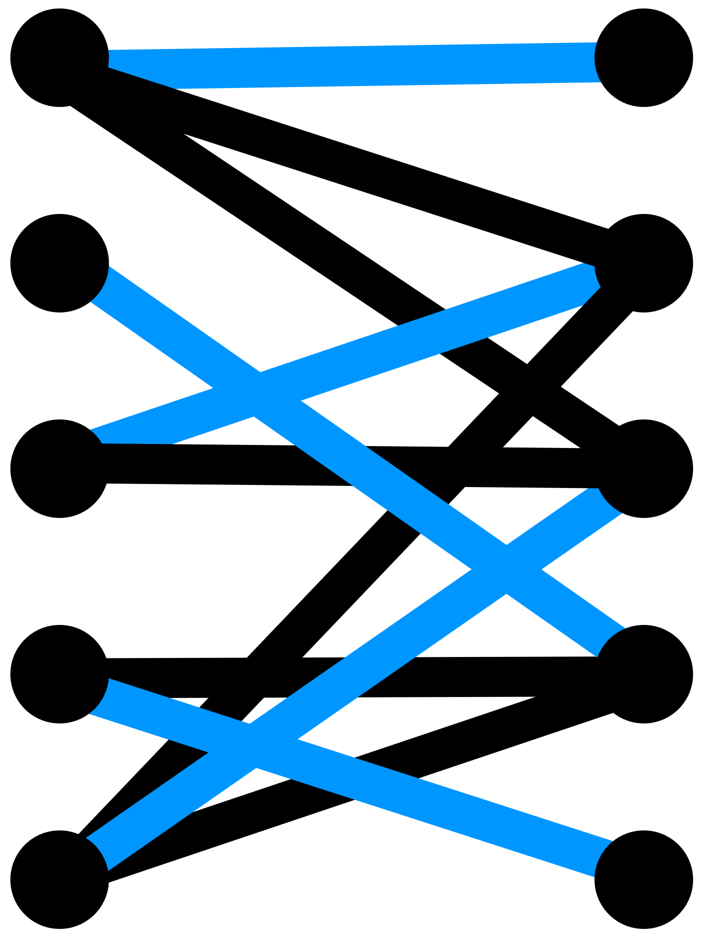
Definition: a matching of a graph is a subset of endpoint-disjoint edges

Papers Overview

Background: Matching Theory



not a max matching



a max matching

Definition: the max matching is the matching with the most edges

Papers Overview

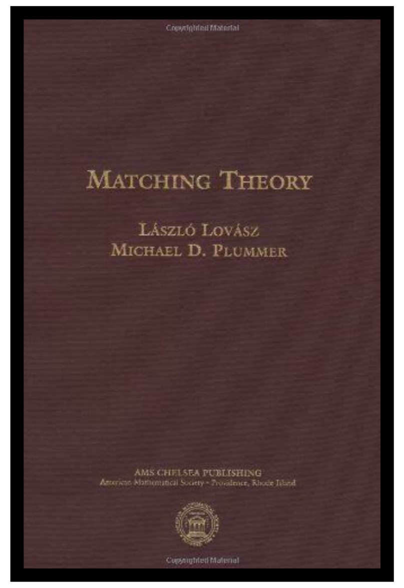
Background: Matching Theory

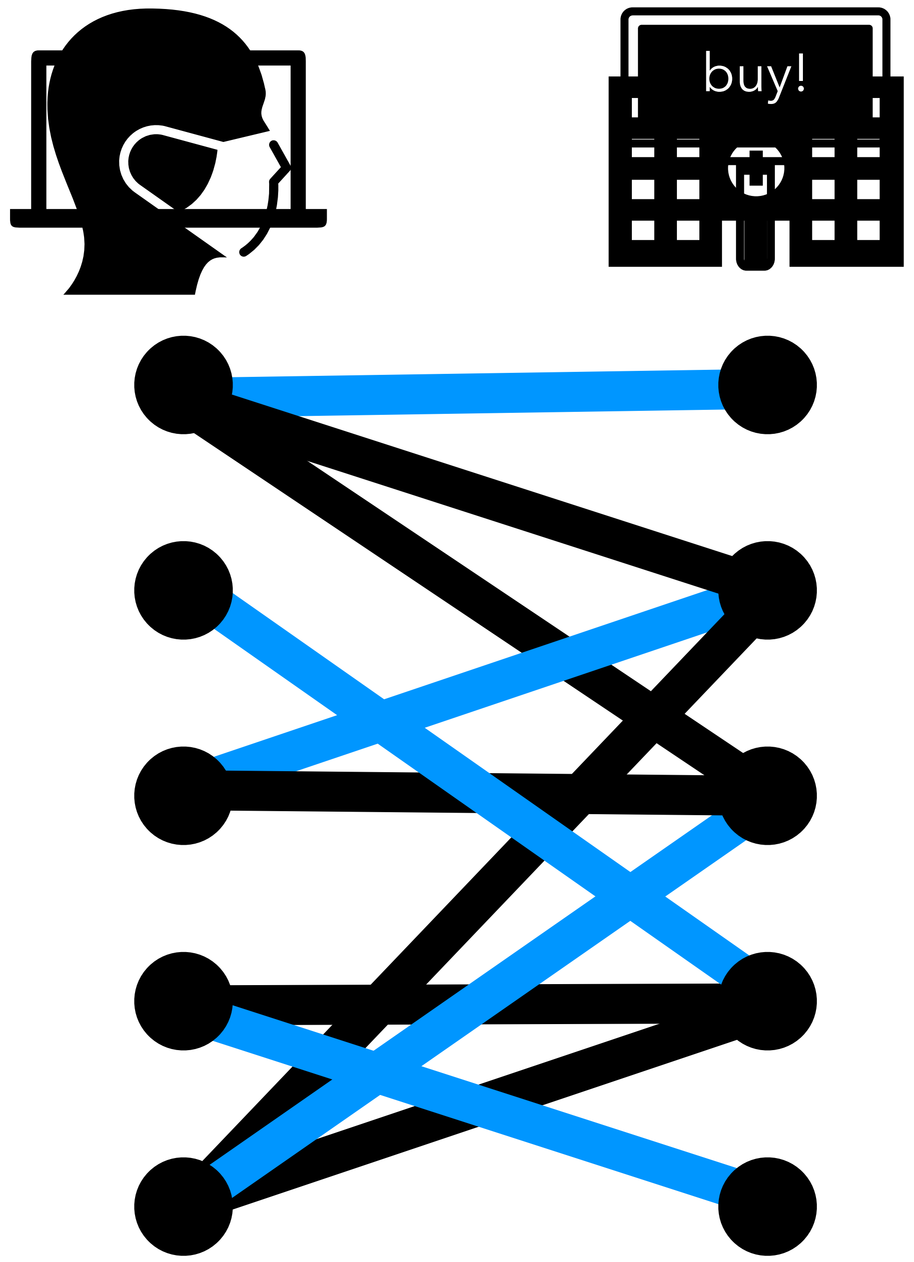
- **Flexible model**

  ads -> users

  doctors -> hospitals

- **Mathematically deep**

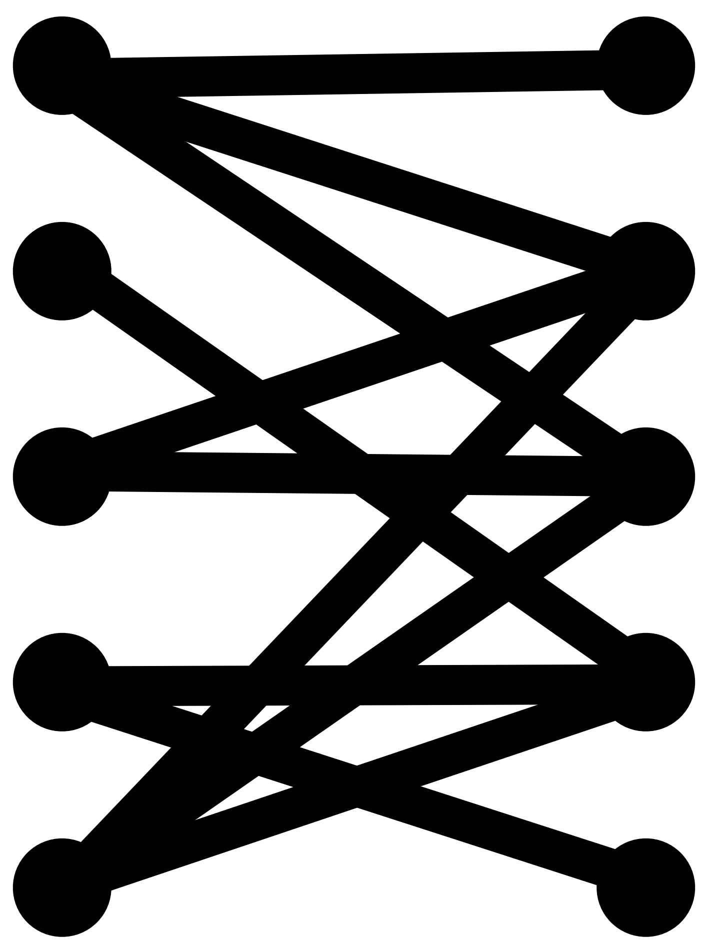




matching

Papers Overview

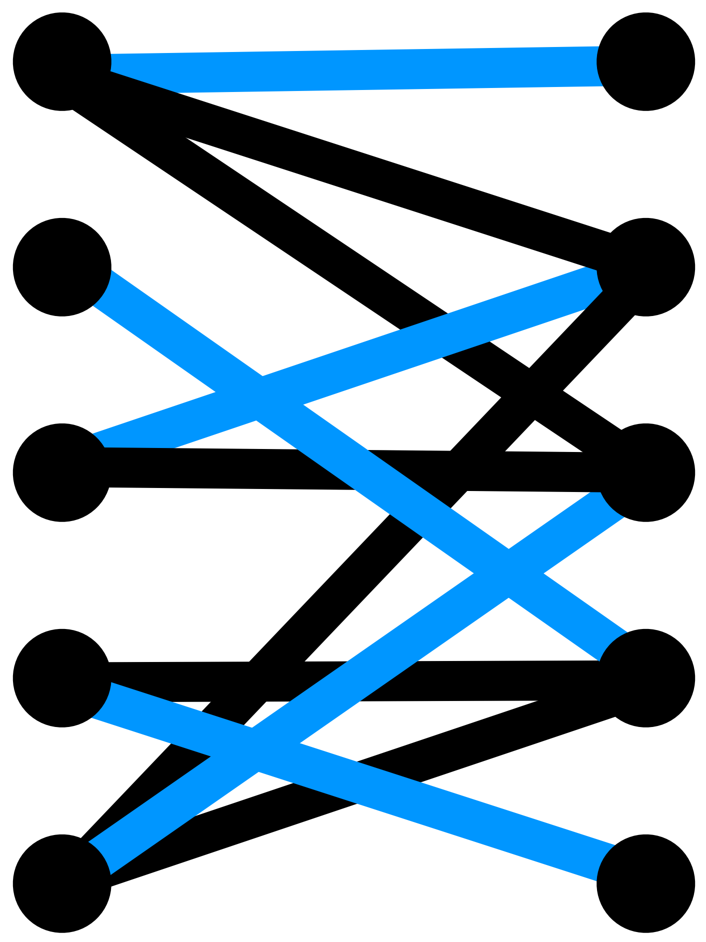
Paper 9: Matching Sparsification



Goal: efficiently maintain near-max-matching dynamically

Papers Overview

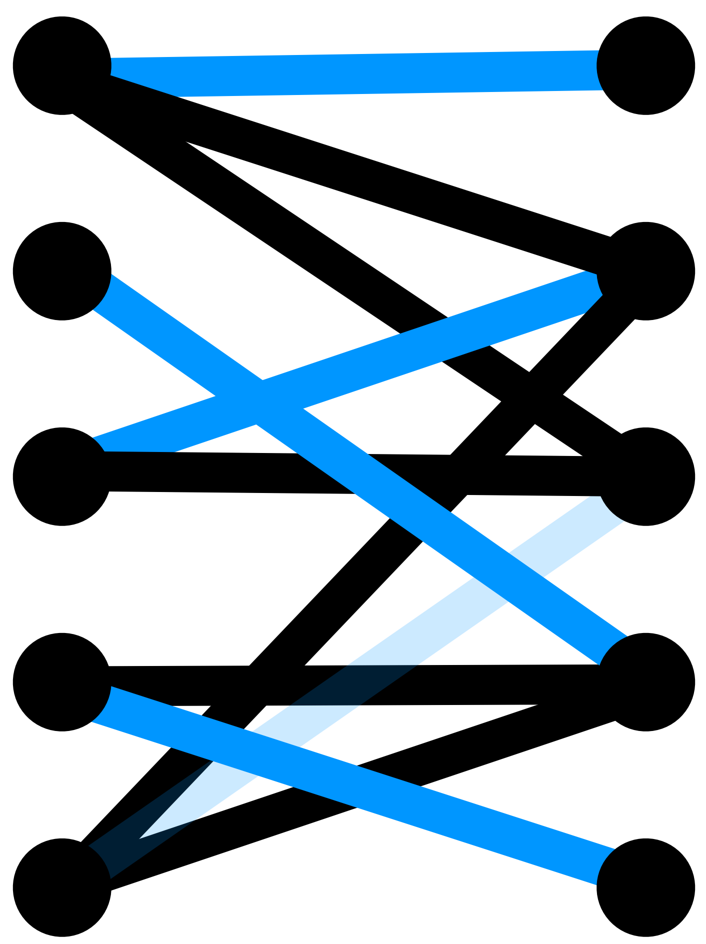
Paper 9: Matching Sparsification



Goal: efficiently maintain near-max-matching dynamically

Papers Overview

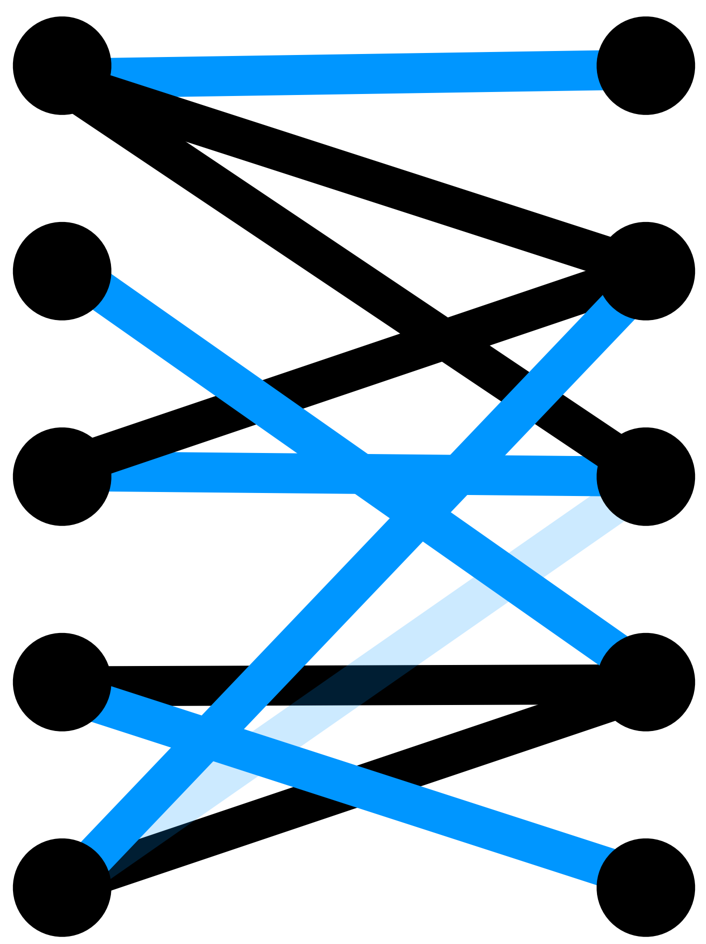
Paper 9: Matching Sparsification



Goal: efficiently maintain near-max-matching dynamically

Papers Overview

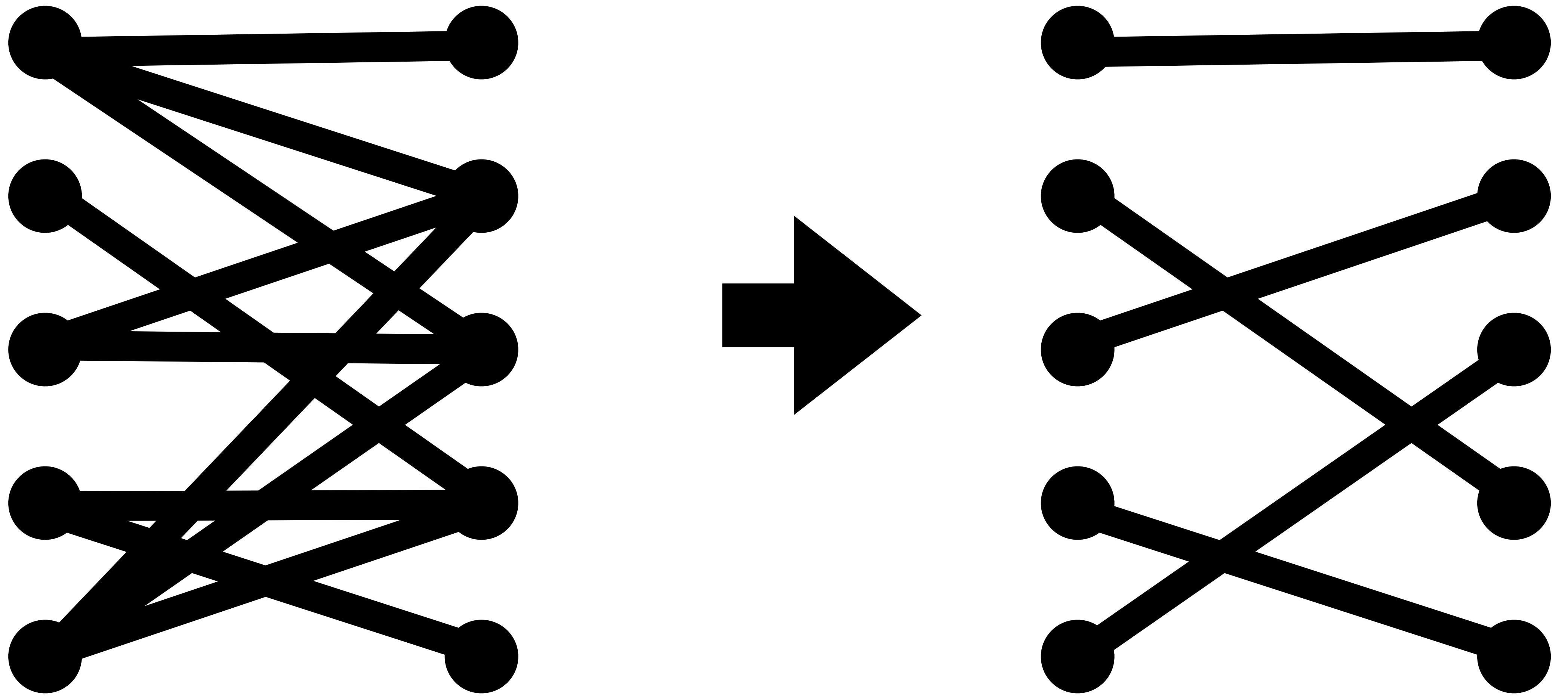
Paper 9: Matching Sparsification



Goal: efficiently maintain near-max-matching dynamically

Papers Overview

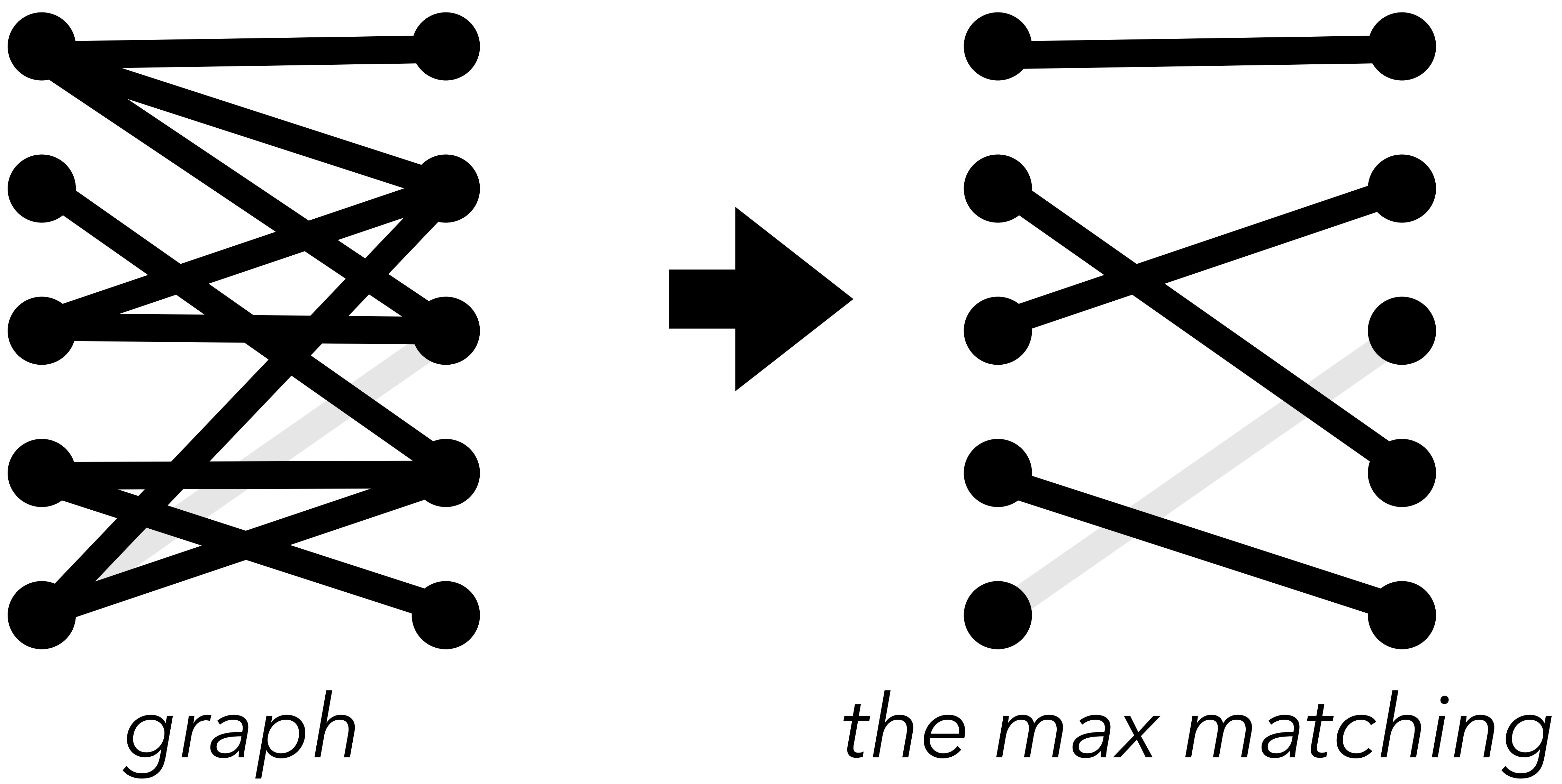
Paper 9: Matching Sparsification



Sub-Goal: a sparse robust subgraph ~preserving the max matching value

Papers Overview

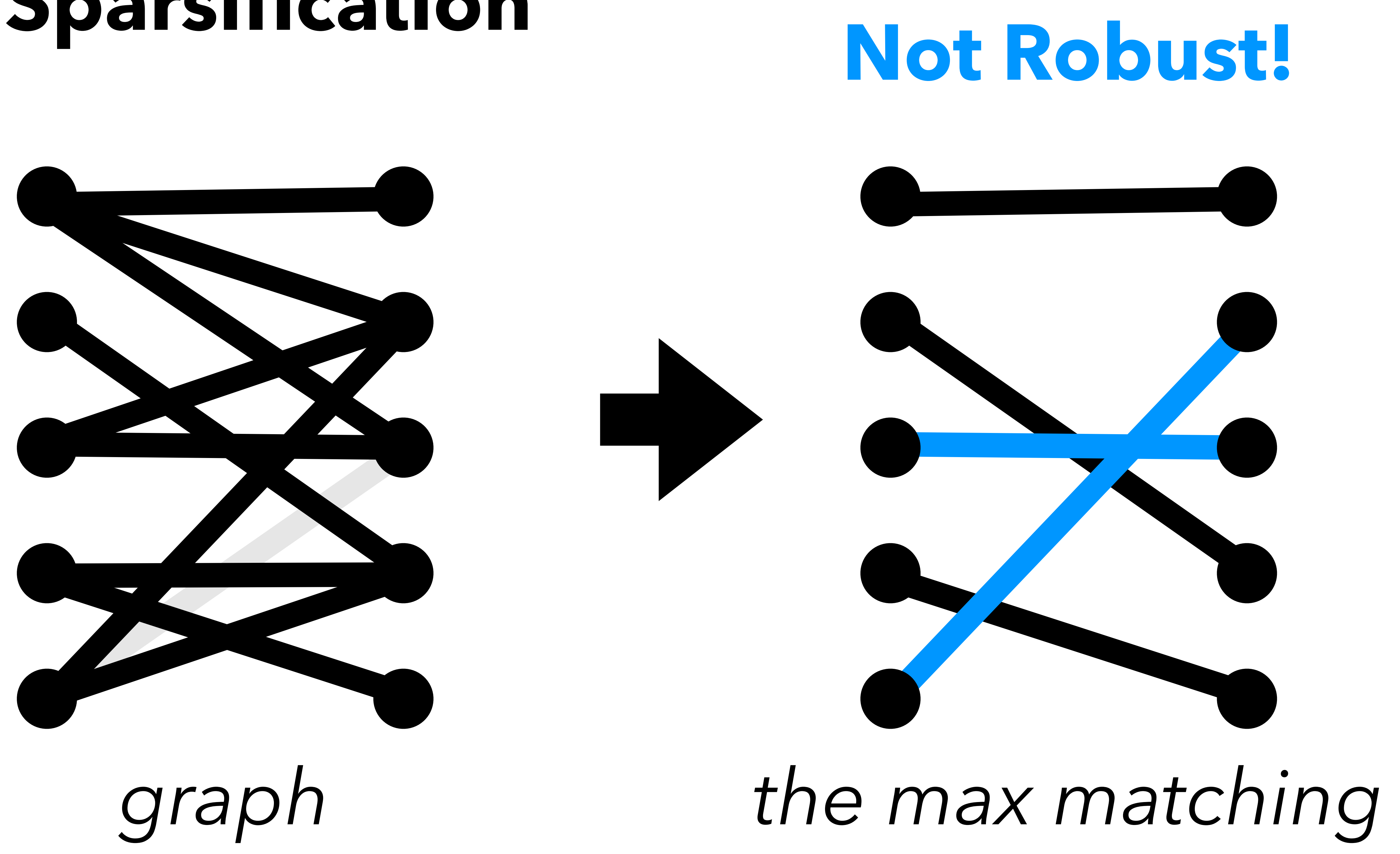
Paper 9: Matching Sparsification



Sub-Goal: a sparse robust subgraph ~preserving the max matching value

Papers Overview

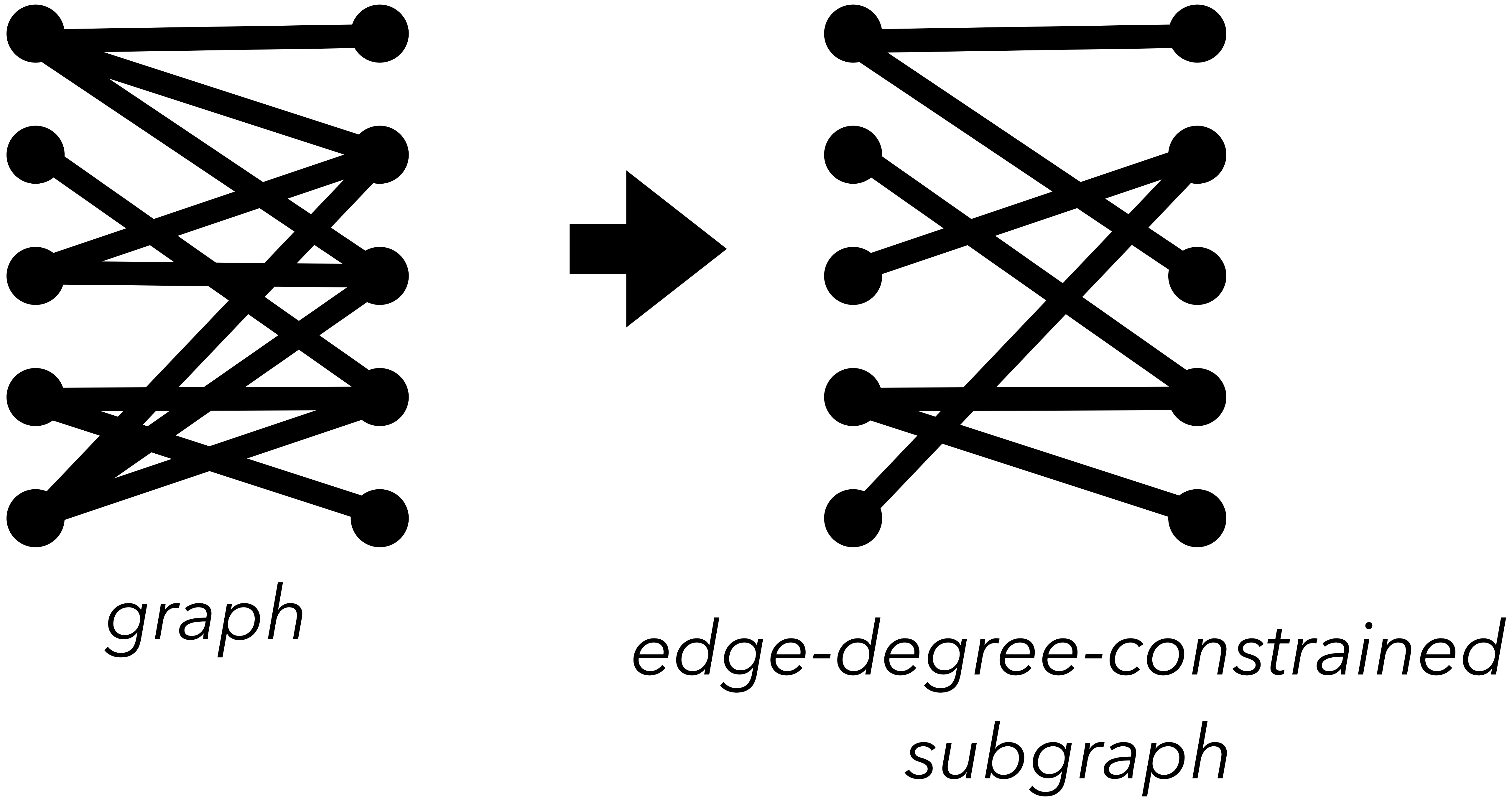
Paper 9: Matching Sparsification



Sub-Goal: a sparse robust subgraph ~preserving the max matching value

Papers Overview

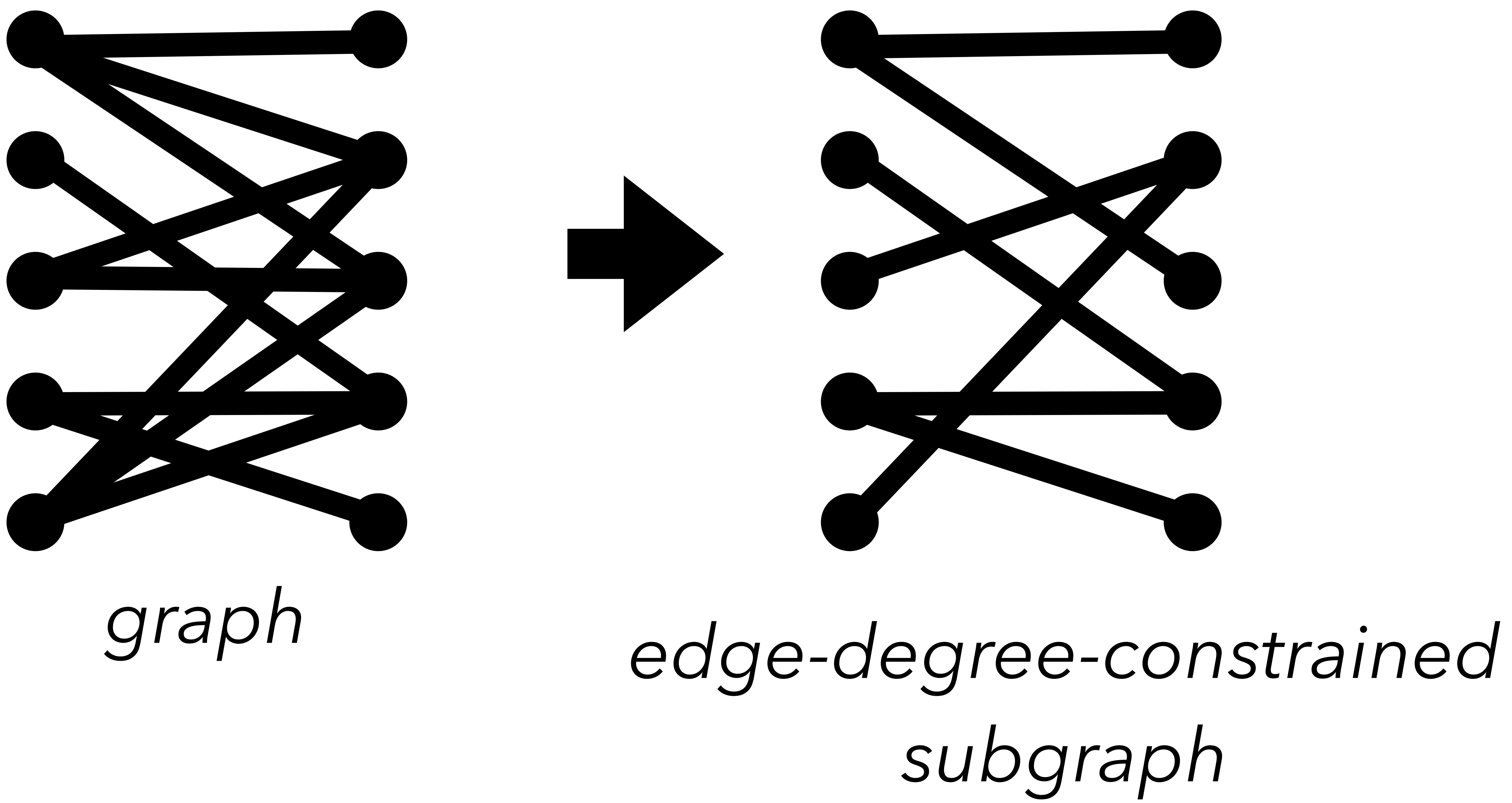
Paper 9: Matching Sparsification



Sub-Goal: a sparse robust subgraph ~preserving the max matching value

Papers Overview

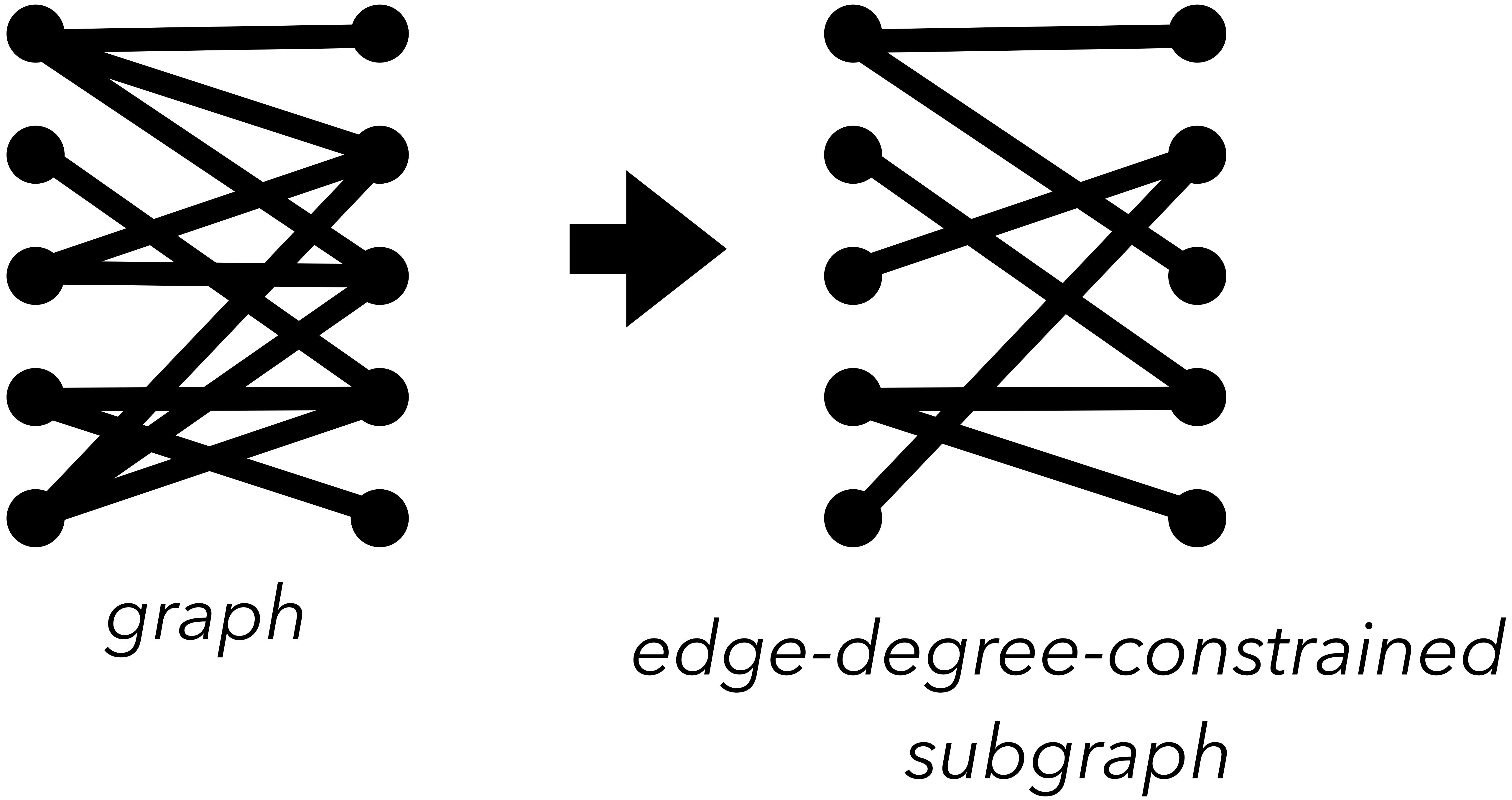
Paper 9: Matching Sparsification



Theorem 1(informal): can maintain a sparse subgraph that $\approx 3/2$ preserves the maximum matching

Papers Overview

Paper 9: Matching Sparsification

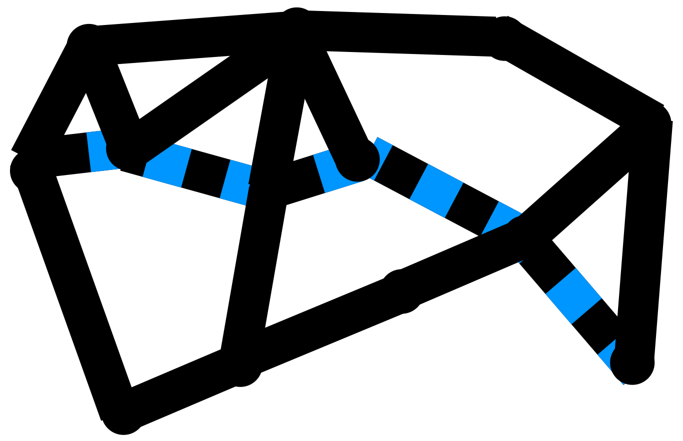


Theorem 2: can maintain a $\approx 3/2$-approximate matching in amortized time $\approx m^{1/4}$ per edge change

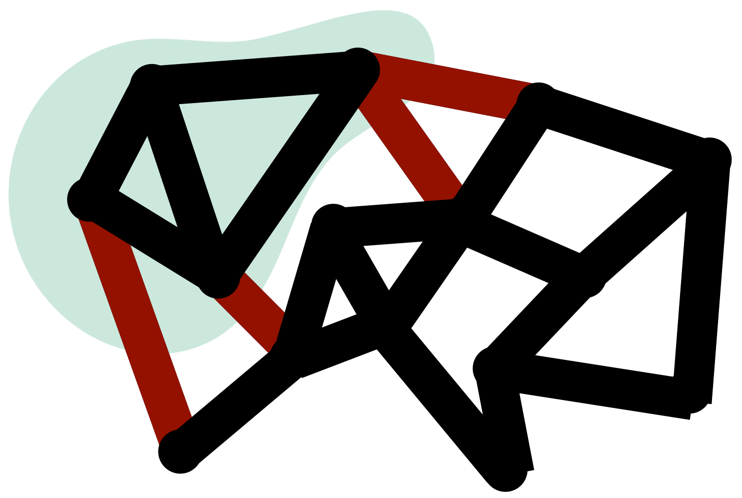
Papers Overview

Sparsification of Five Graph-Theoretic Objects

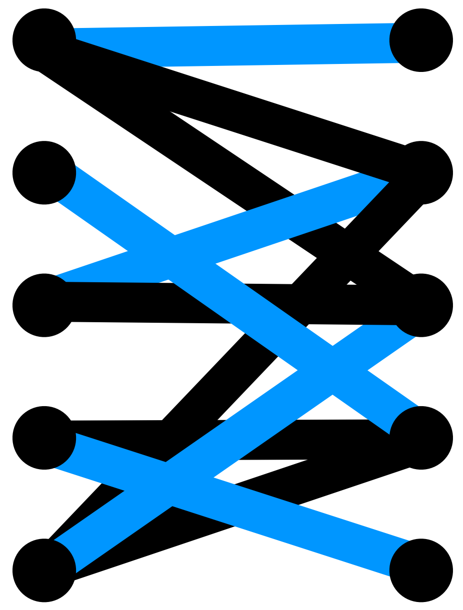
Distances



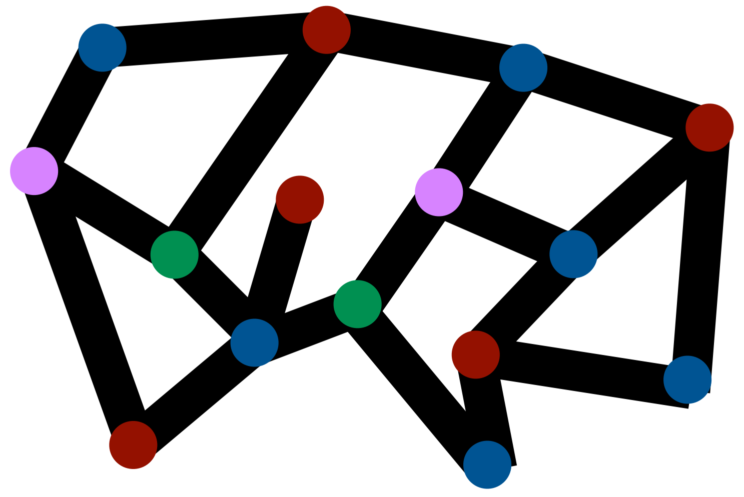
Cuts/Flows



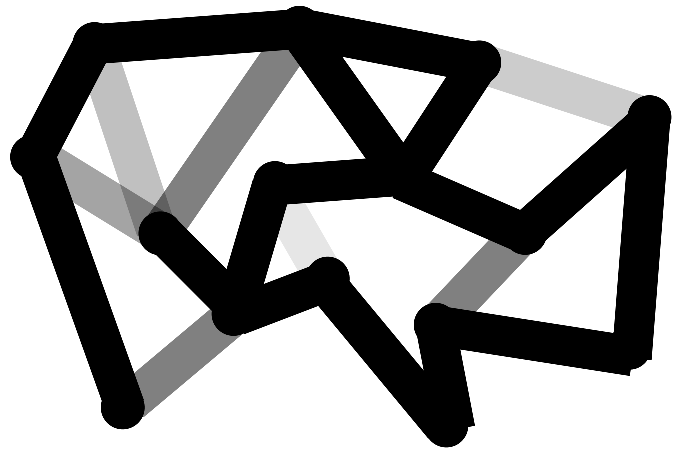
Matchings



Colorings



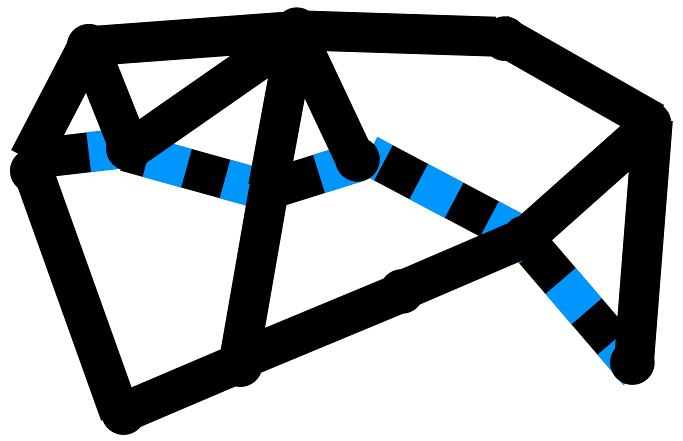
Fractional Opts



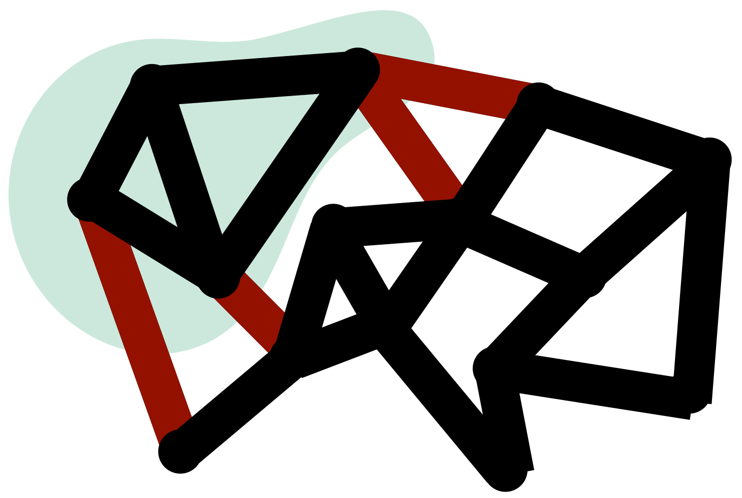
Papers Overview

Sparsification of Five Graph-Theoretic Objects

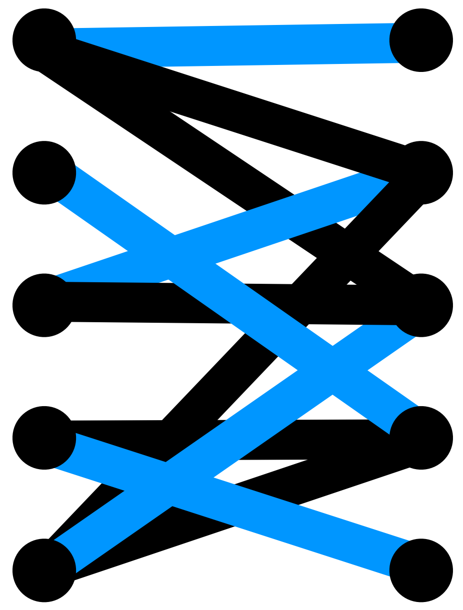
Distances



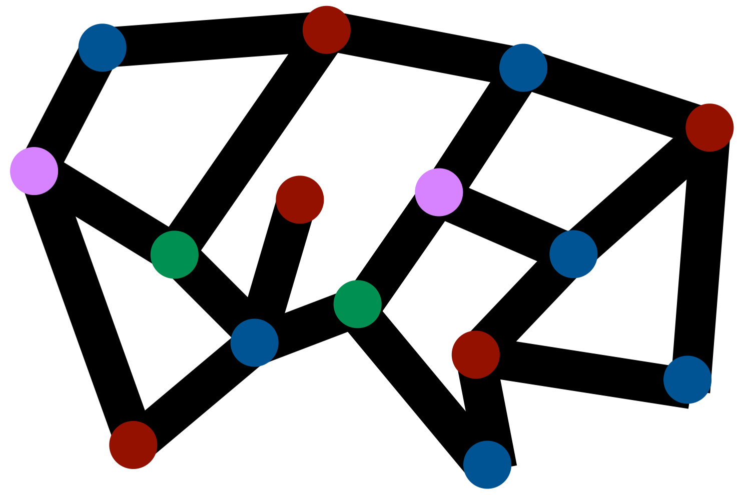
Cuts/Flows



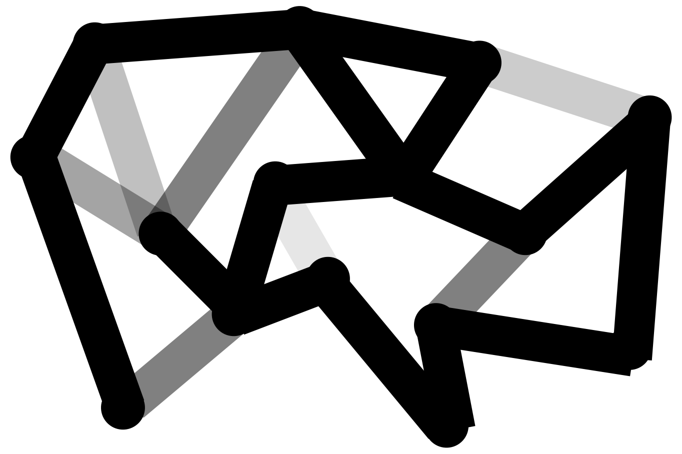
Matchings



Colorings

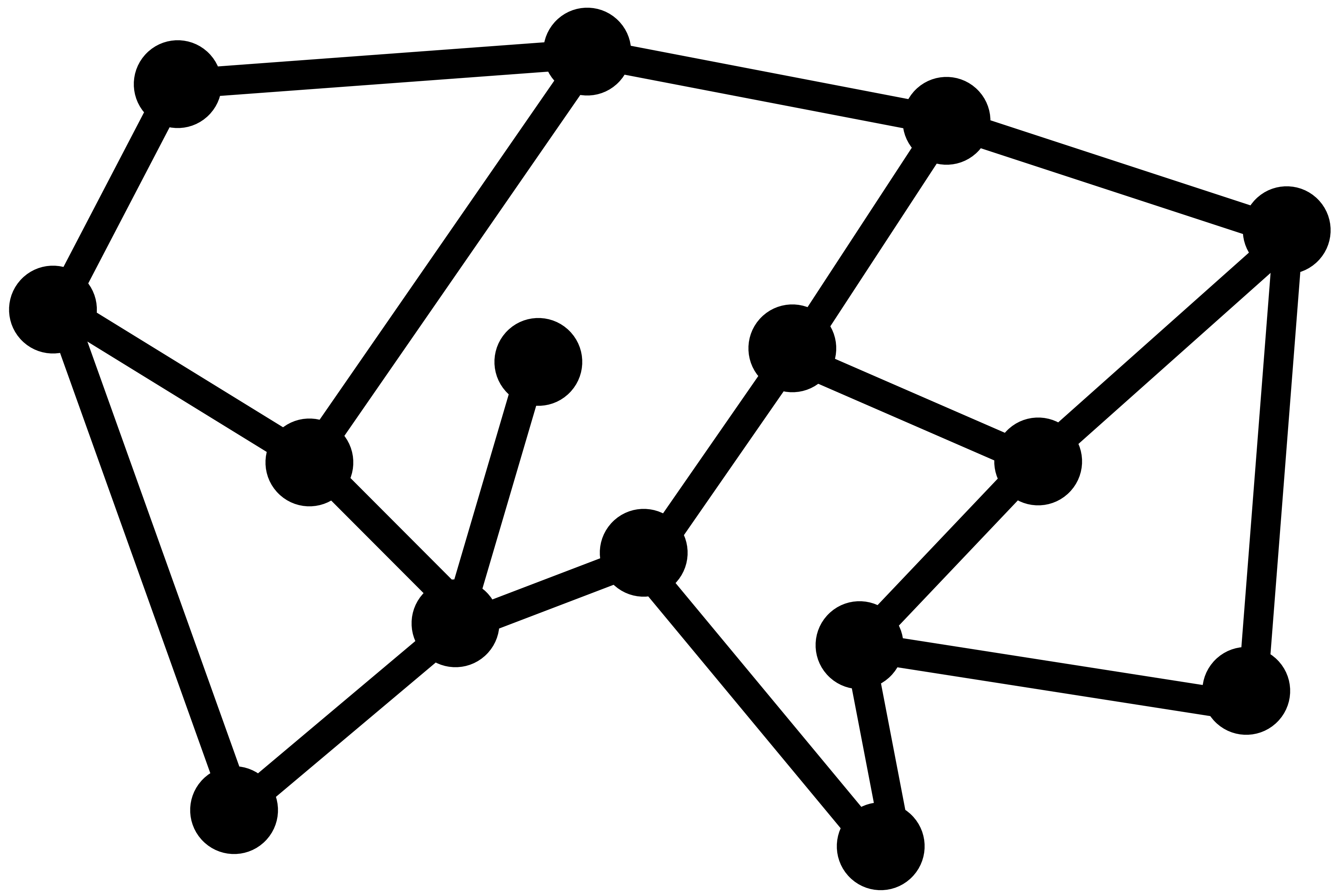


Fractional Opts



Papers Overview

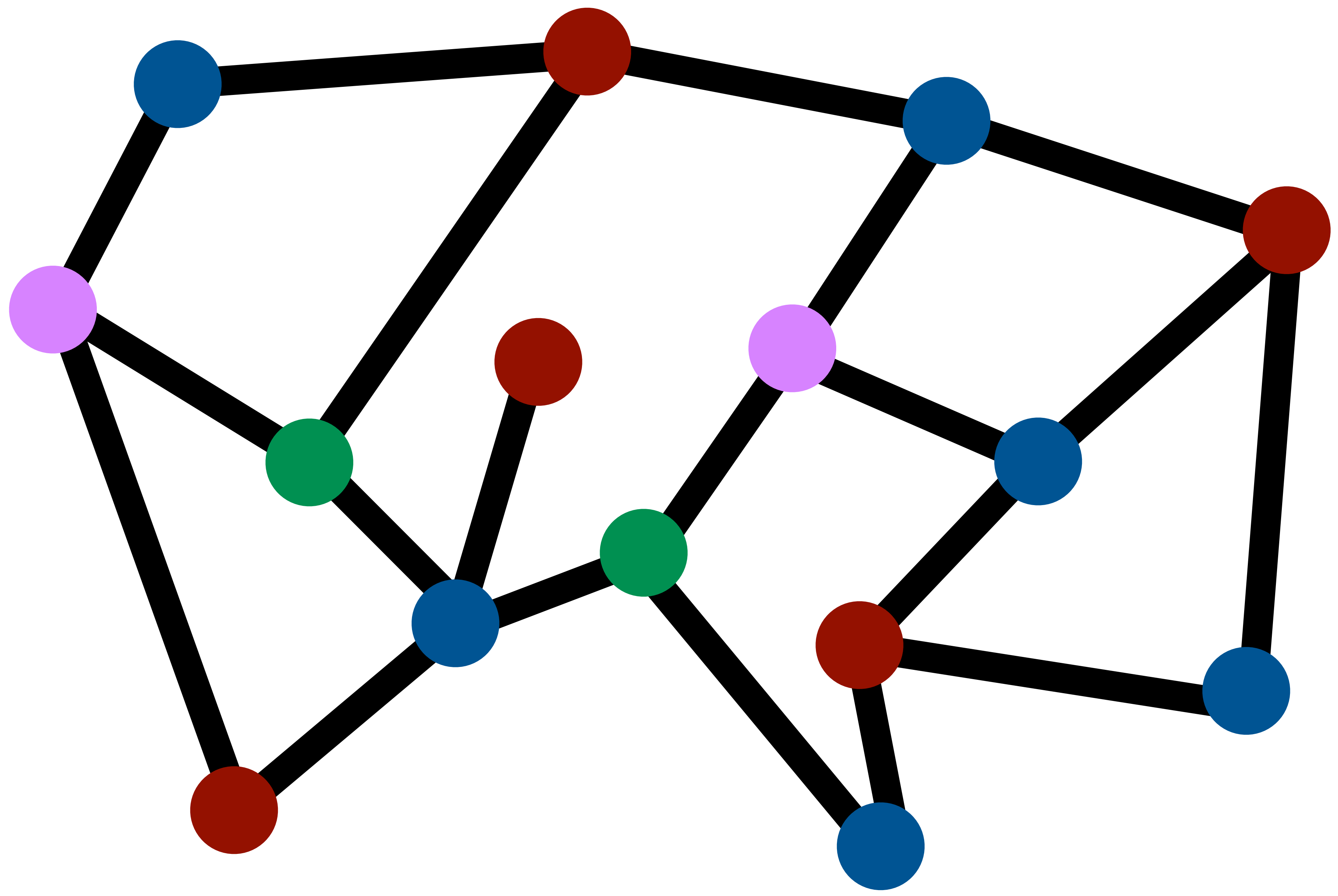
Background: Graph Colorings



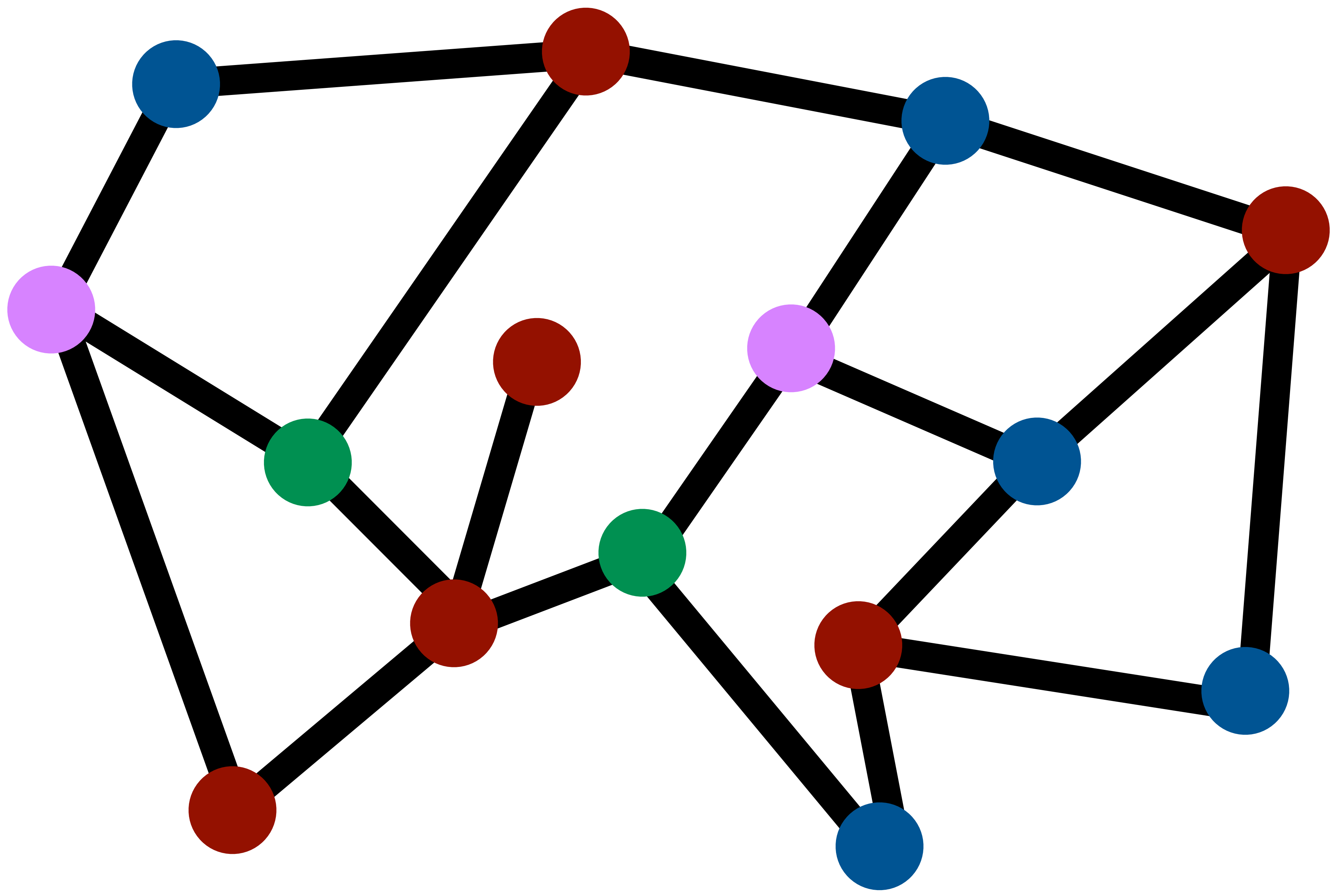
Definition (coloring)**:** a coloring of a graph is an assignment of colors to vertices such that no edge has 2 of the same color

Papers Overview

Background: Graph Colorings



coloring



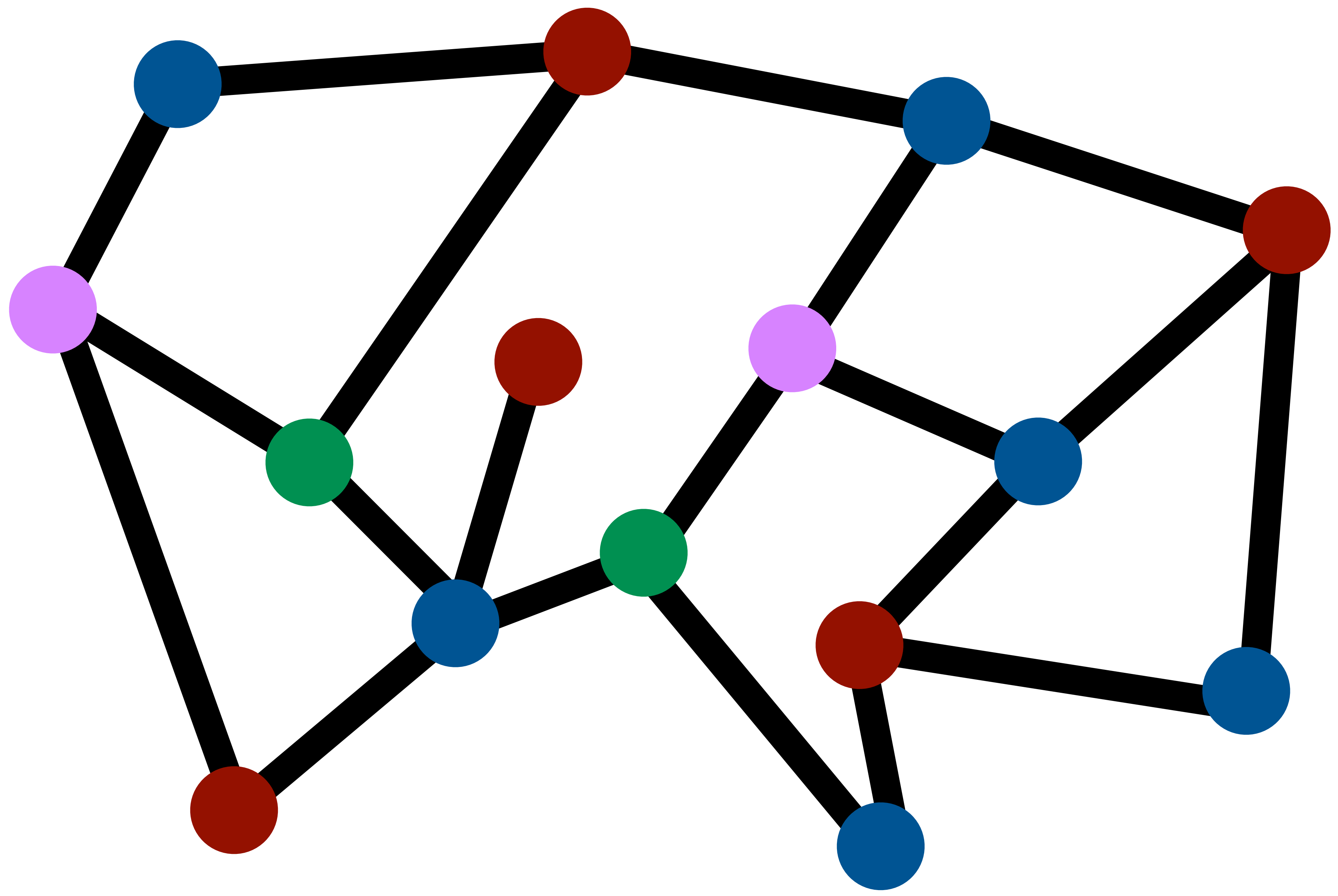
not a coloring

Definition (coloring)**:** a coloring of a graph is an assignment of colors to vertices such that no edge has 2 of the same color

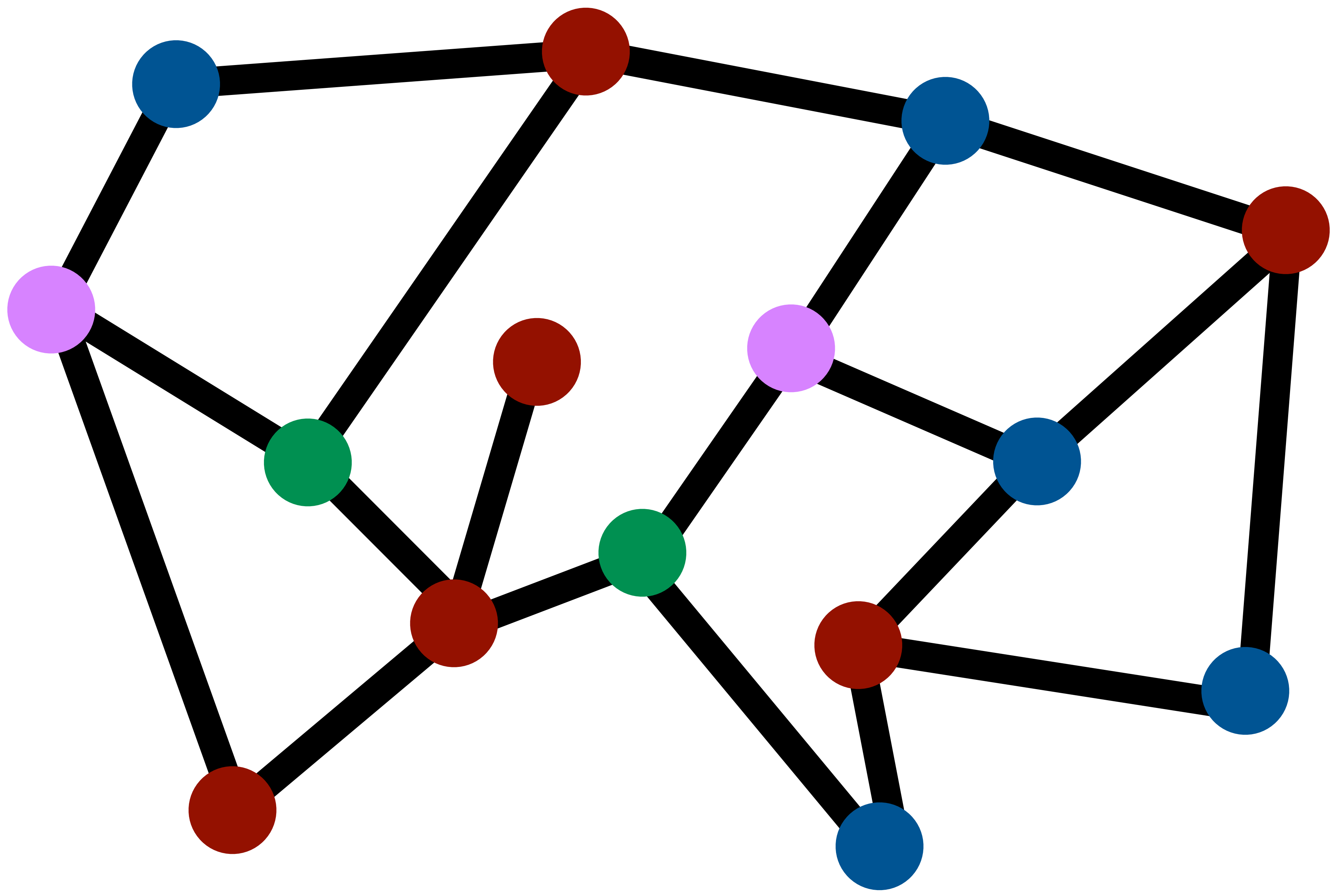
Δ = max degree

Papers Overview

Background: Graph Colorings



coloring



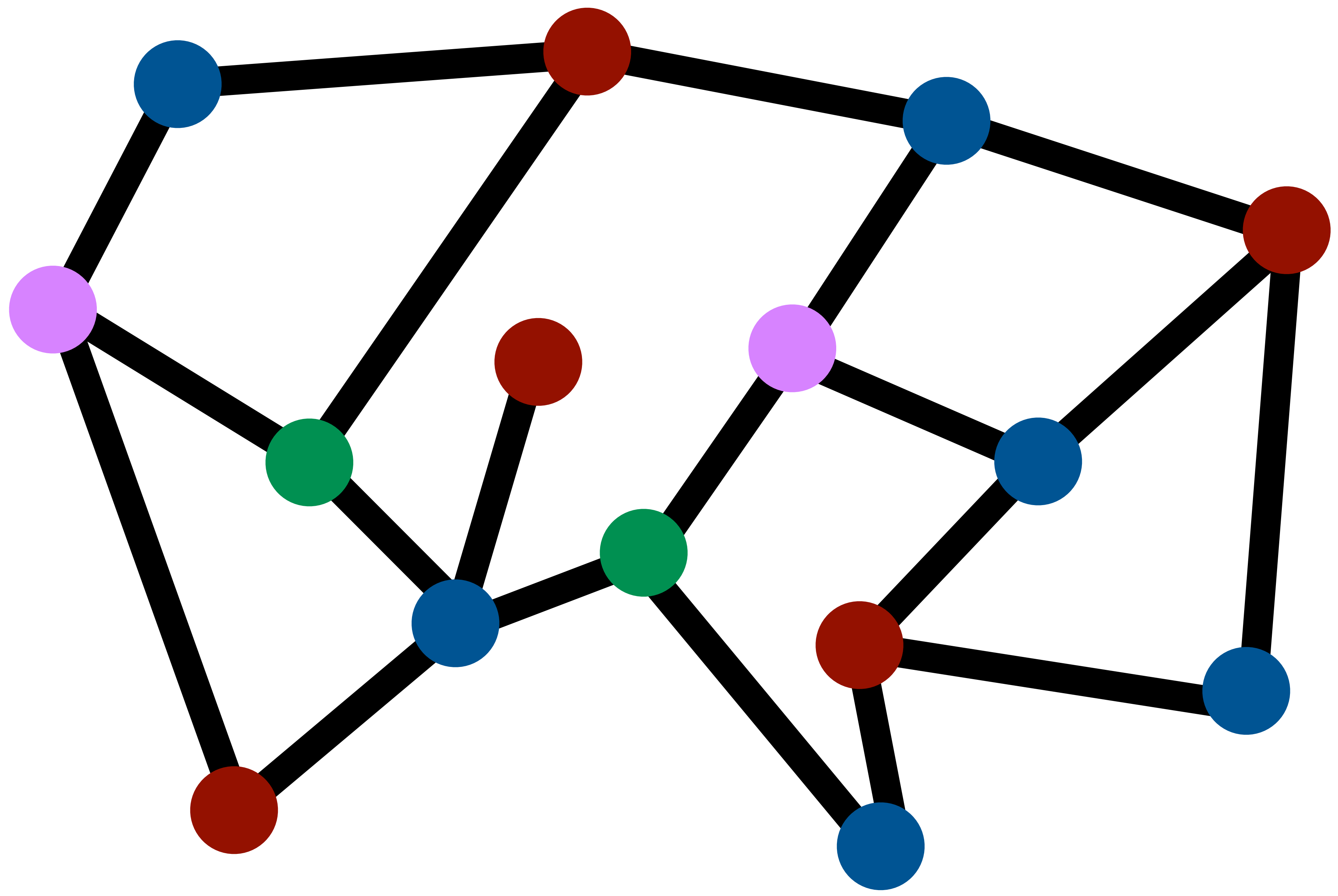
not a coloring

Theorem (folklore): every graph has a $\Delta + 1$ coloring

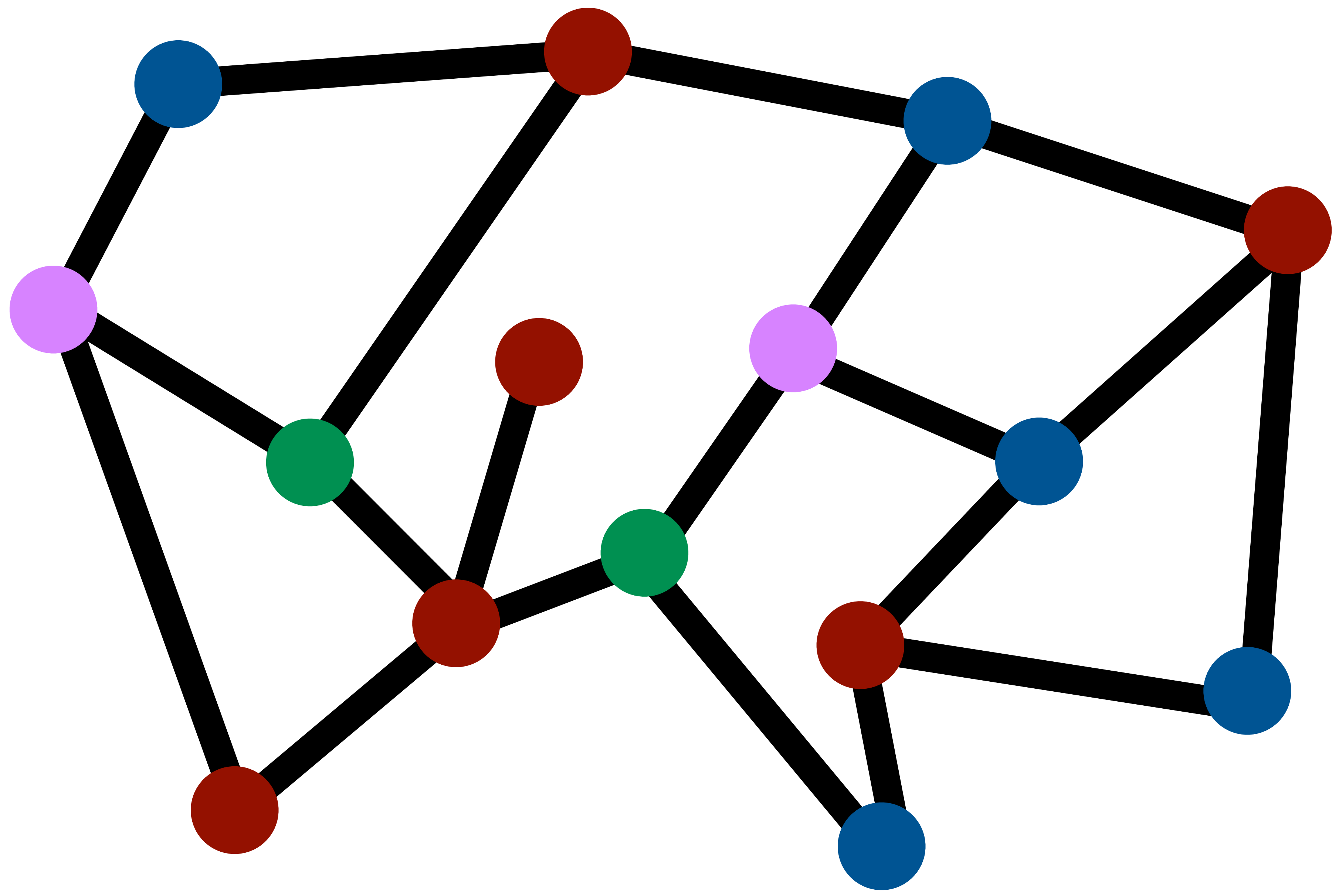
Δ = max degree

Papers Overview

Background: Graph Colorings



coloring



not a coloring

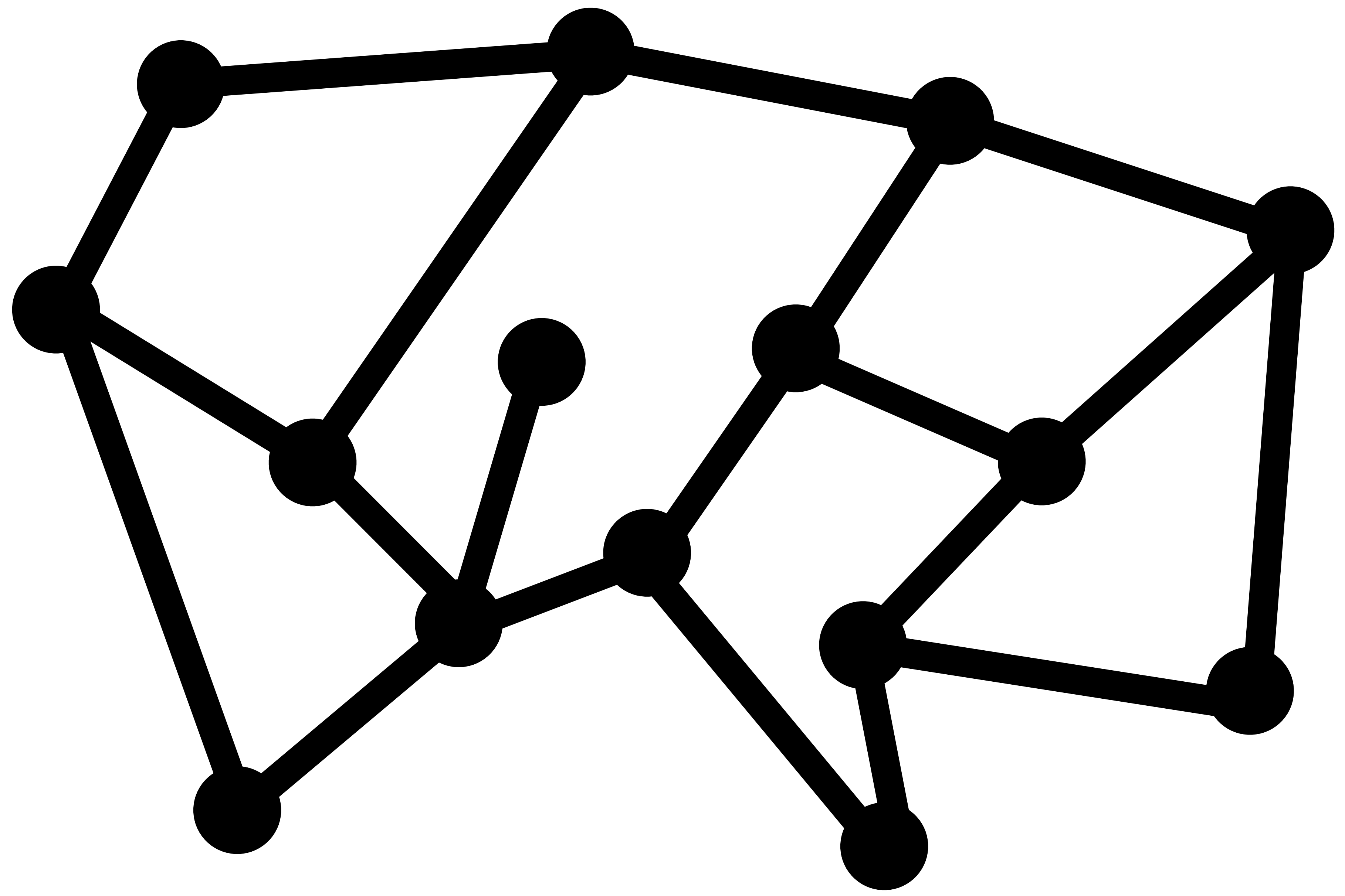
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



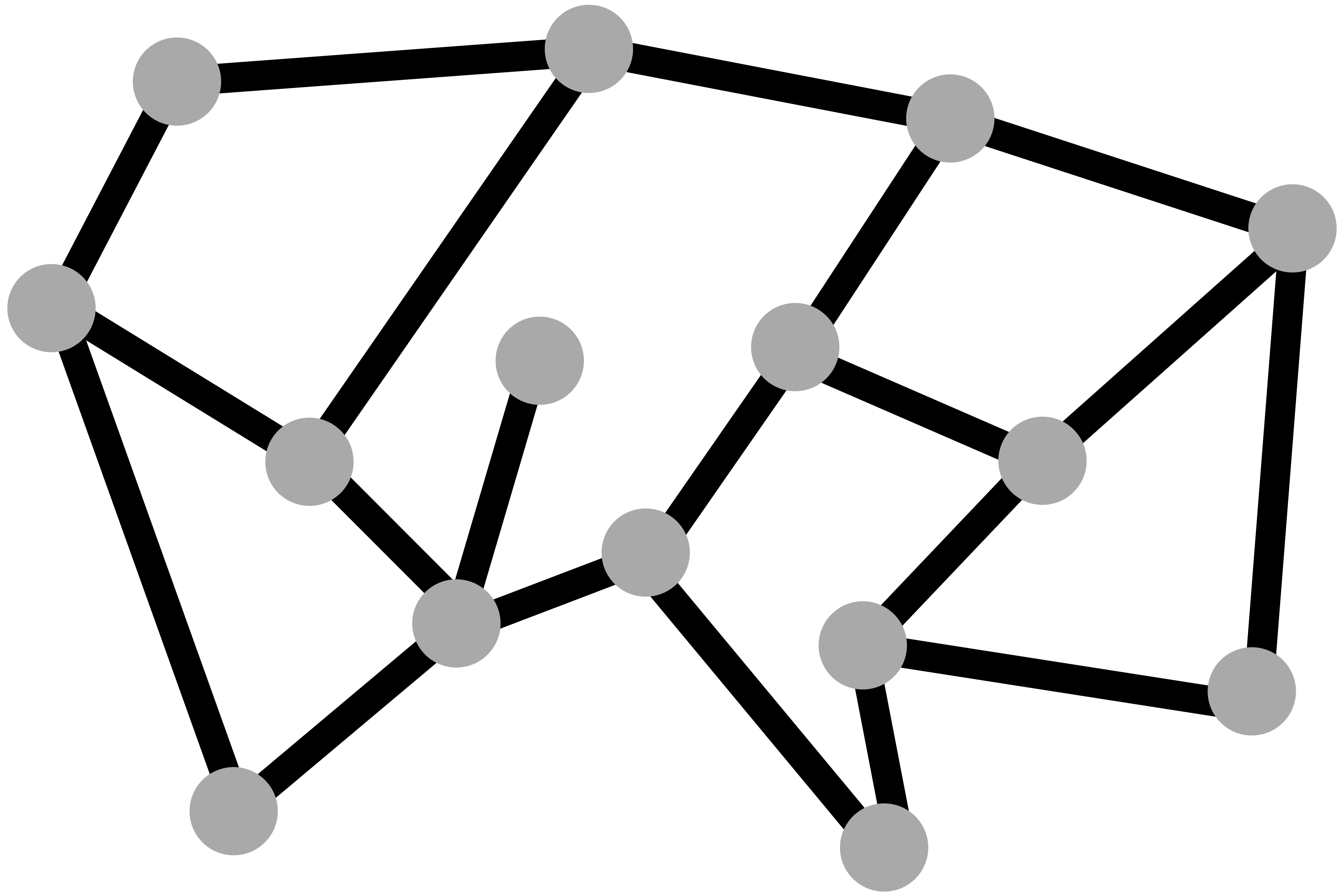
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



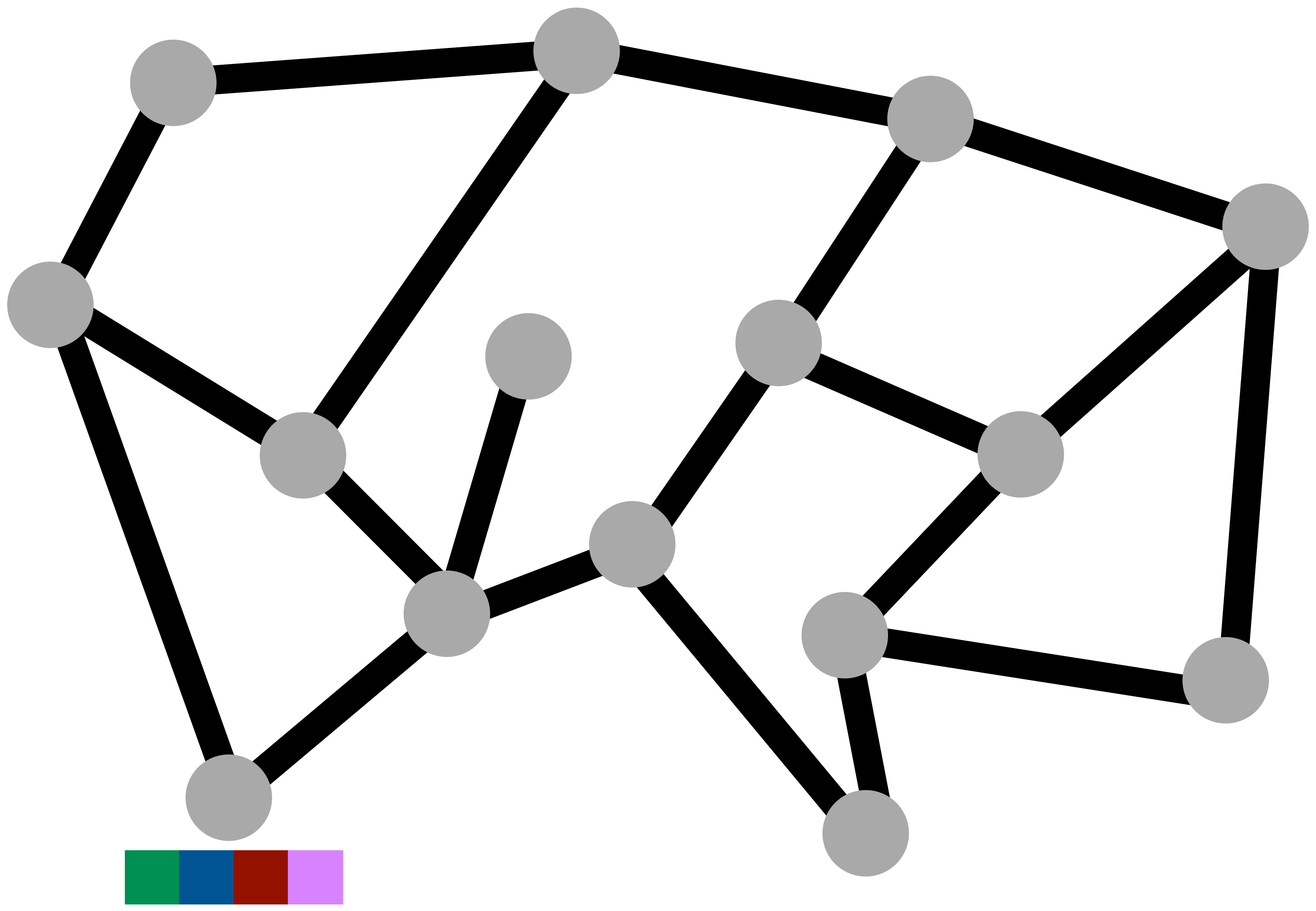
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



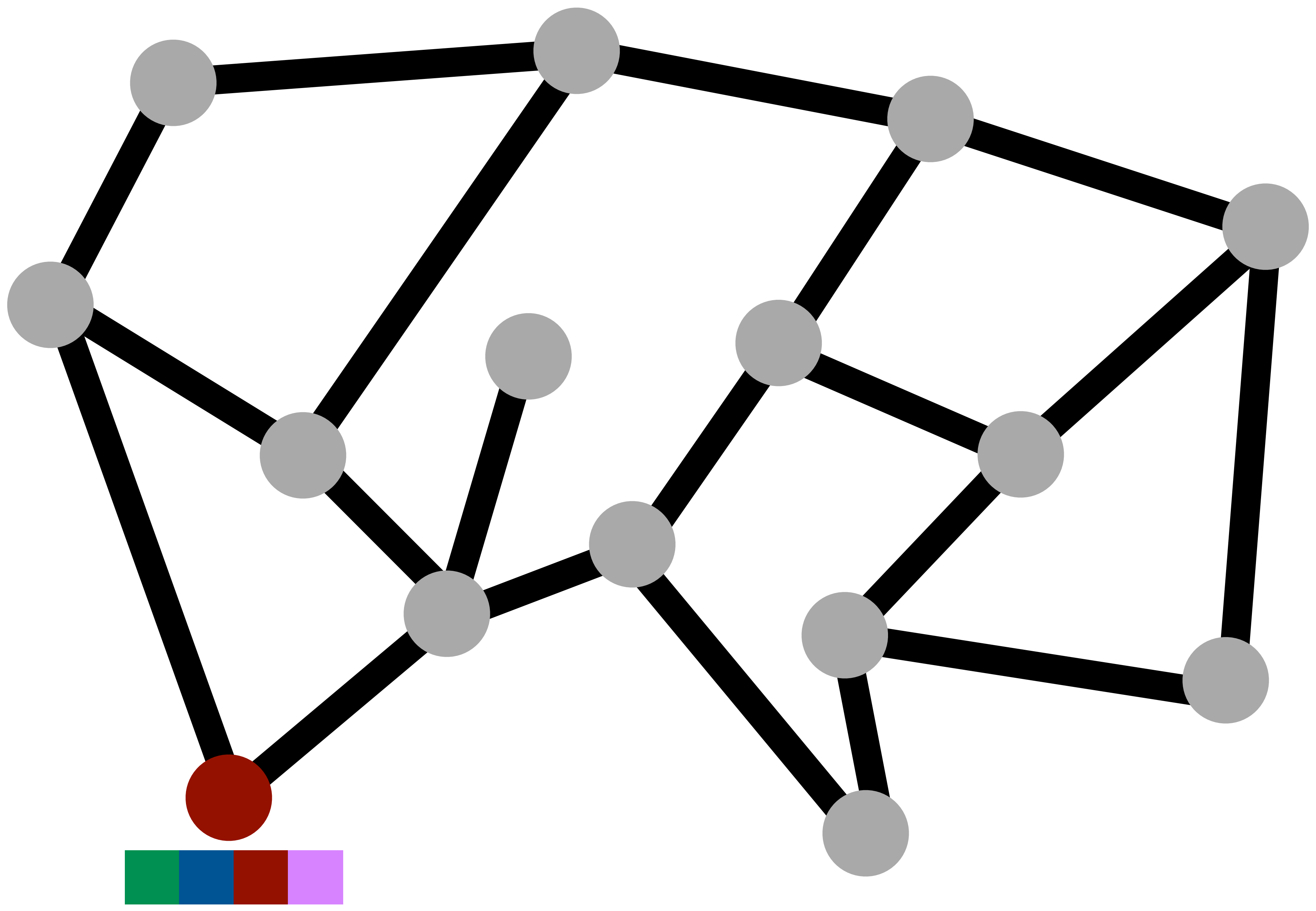
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



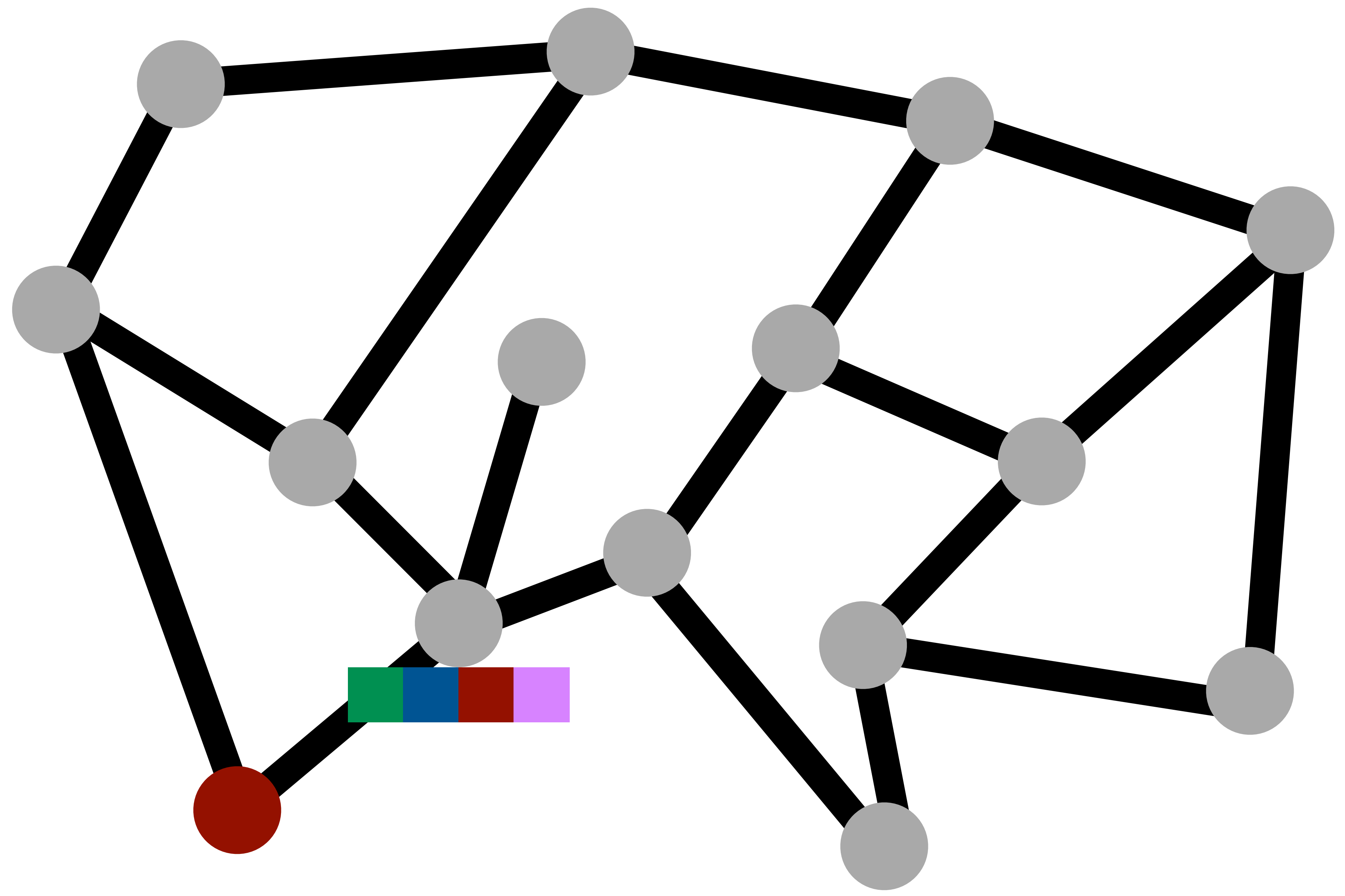
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



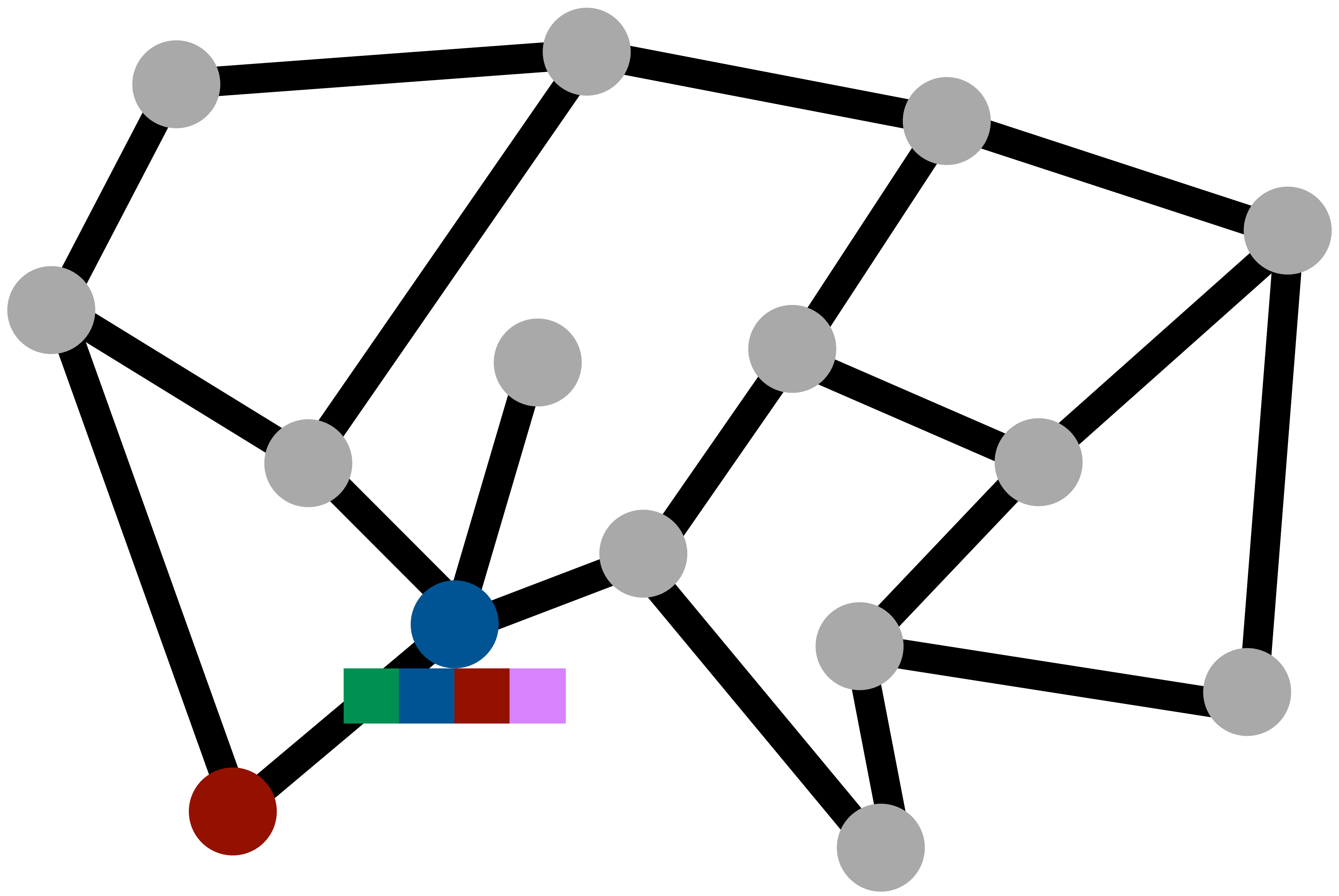
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



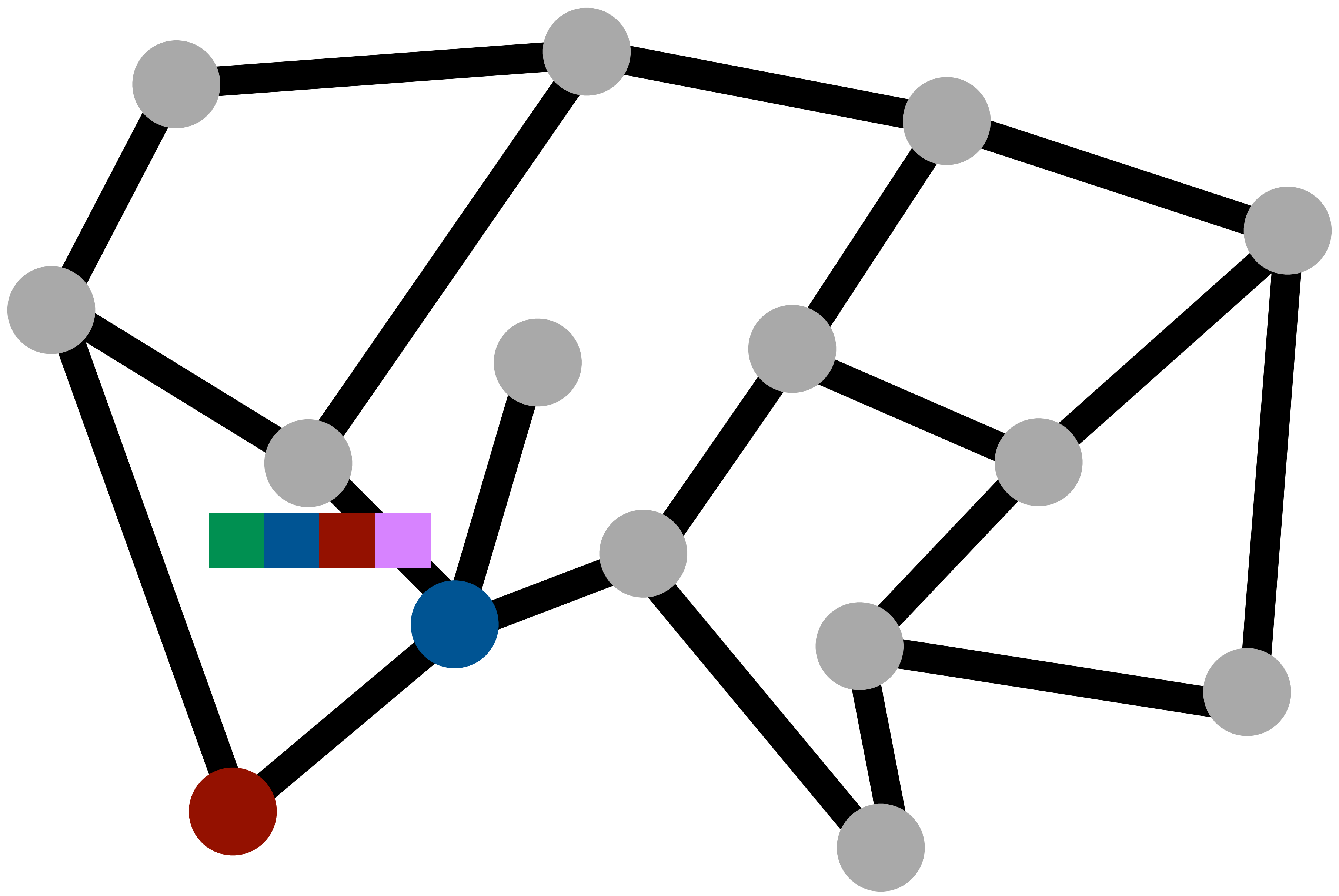
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



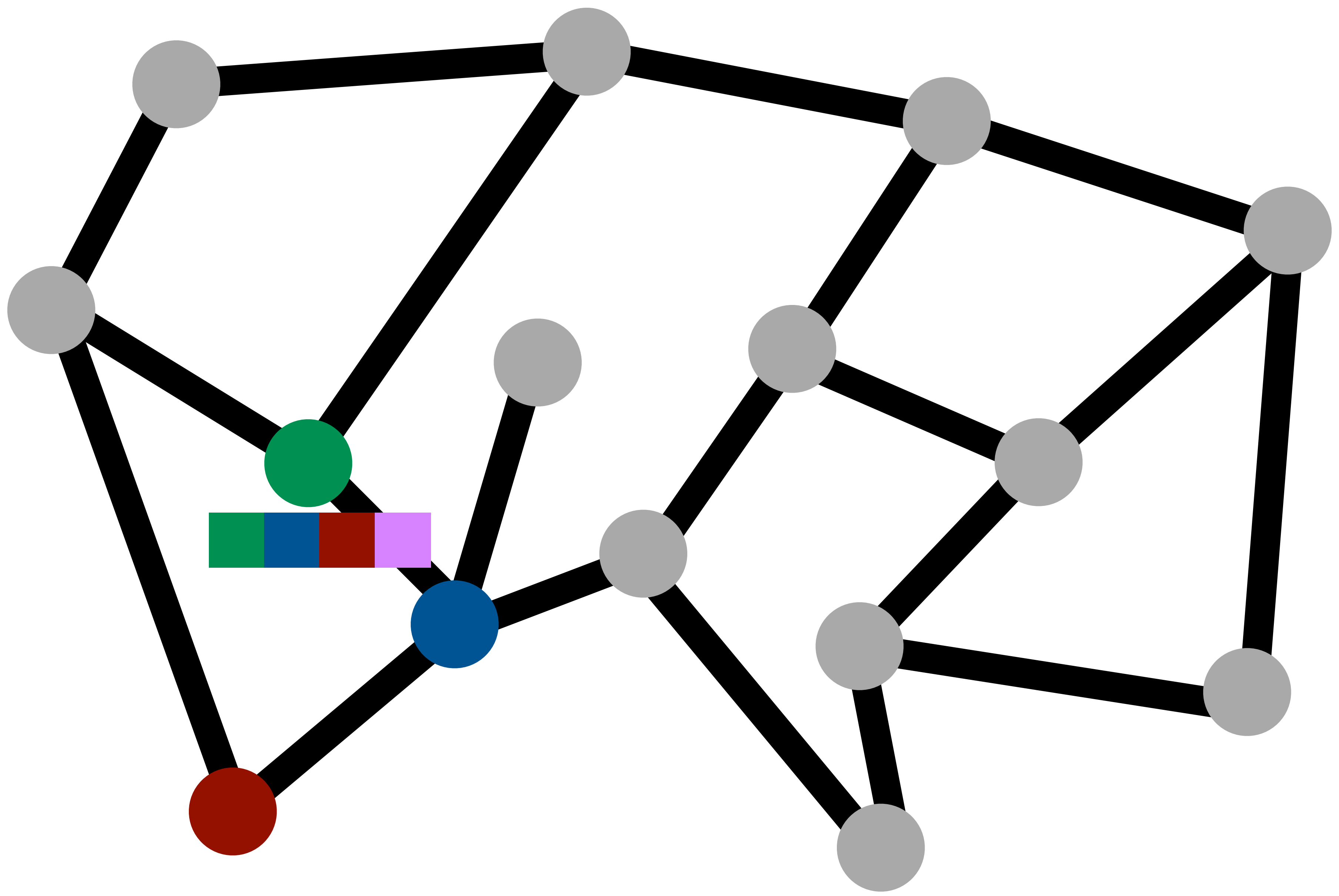
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



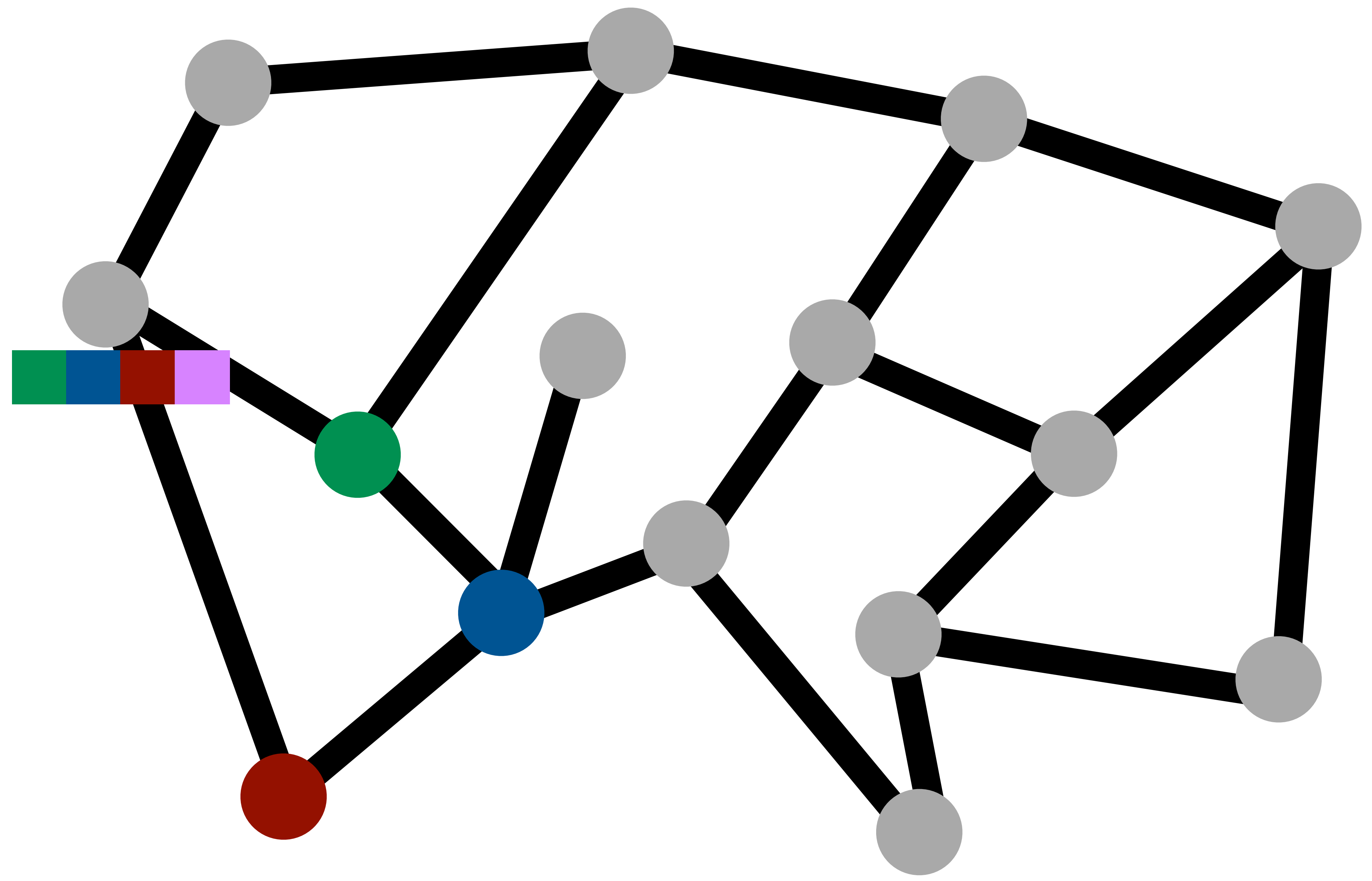
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



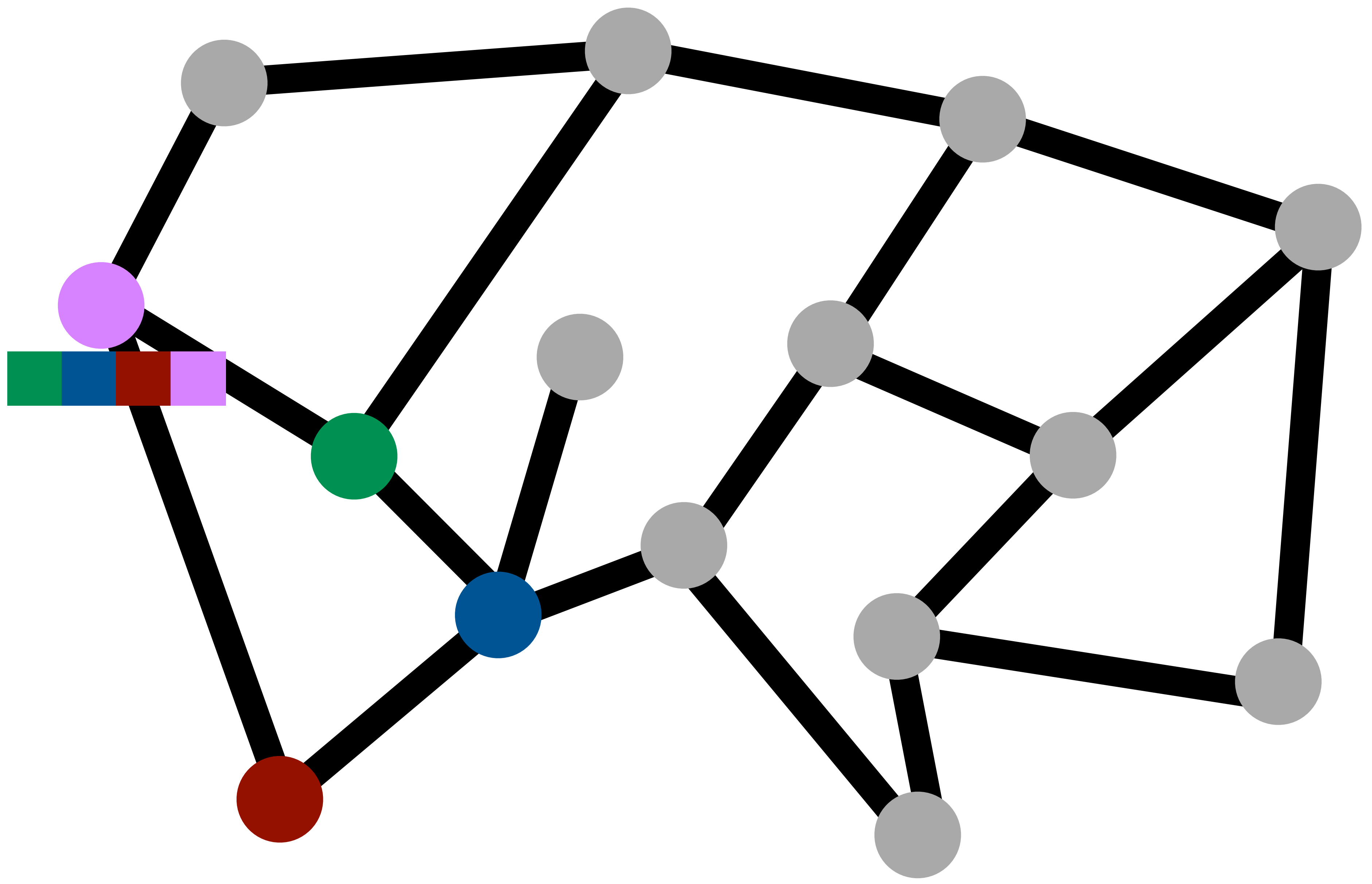
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



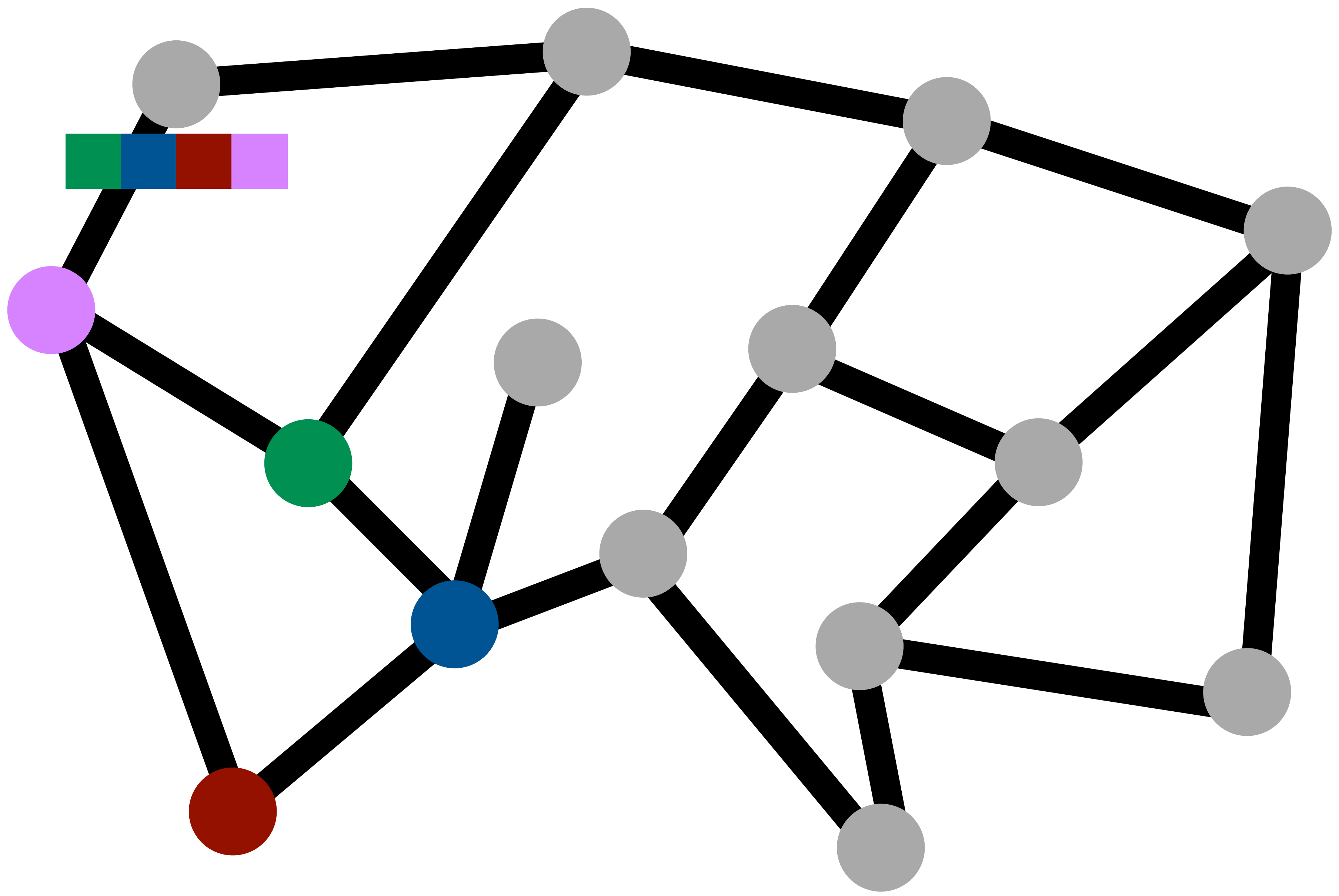
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



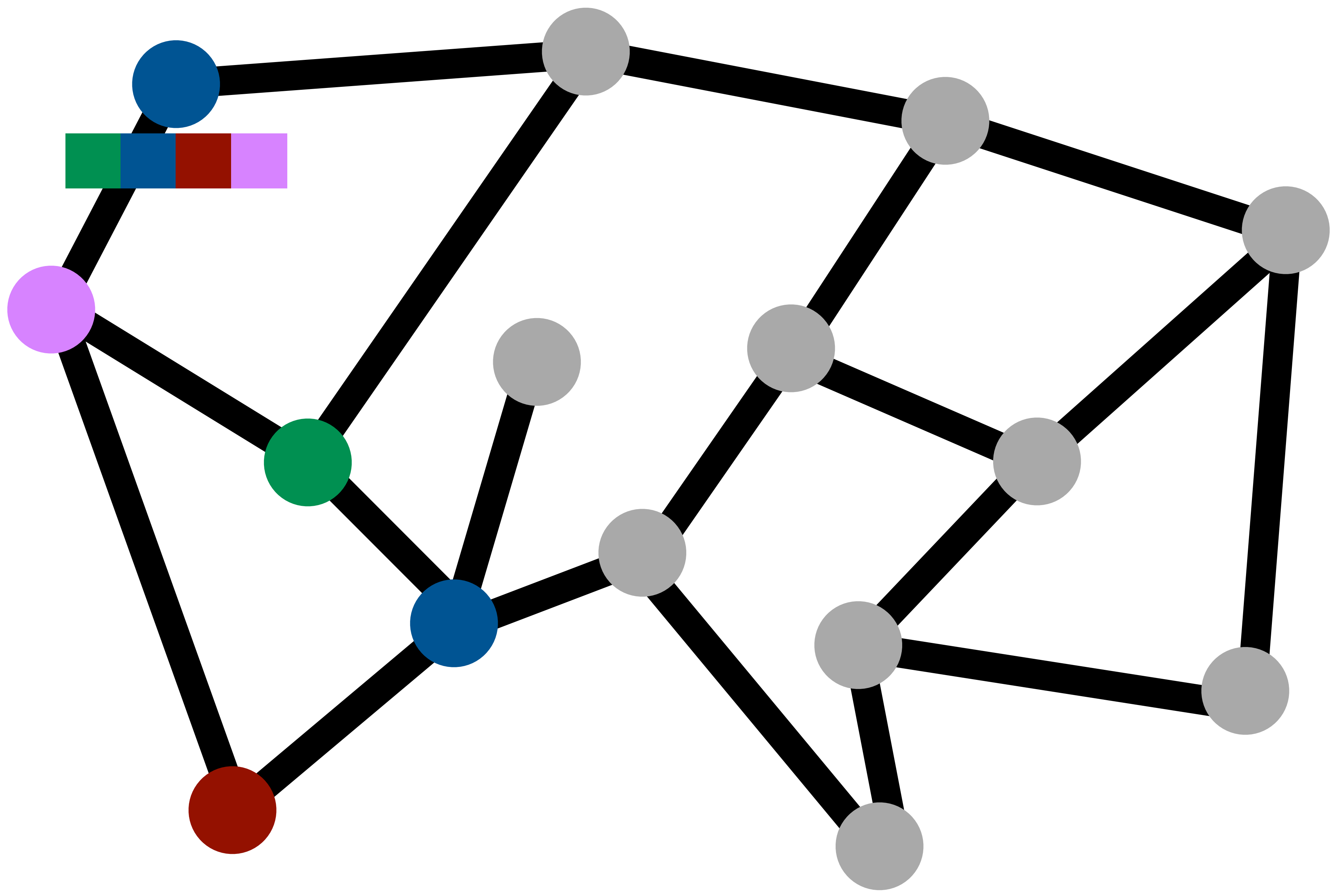
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



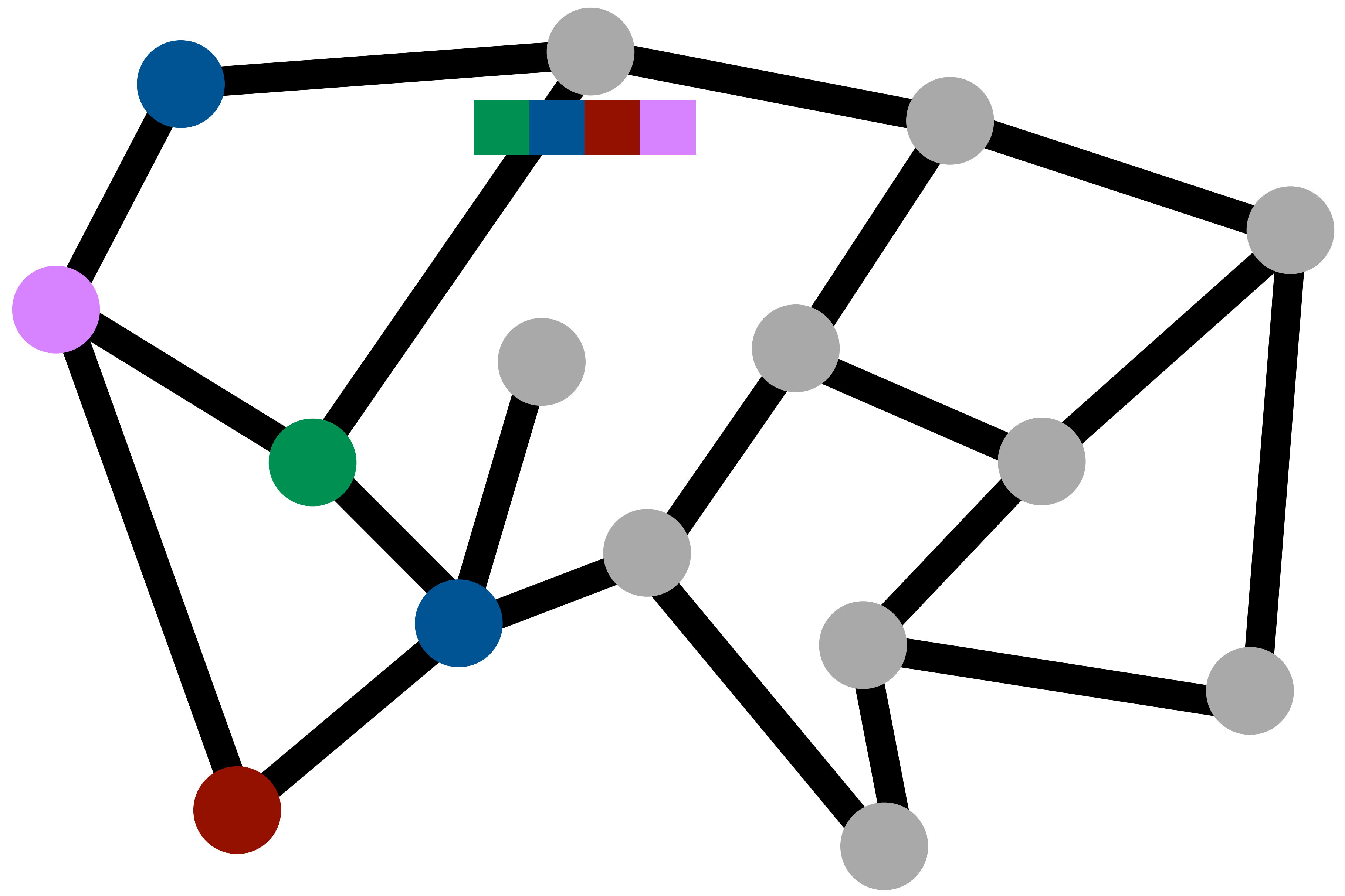
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



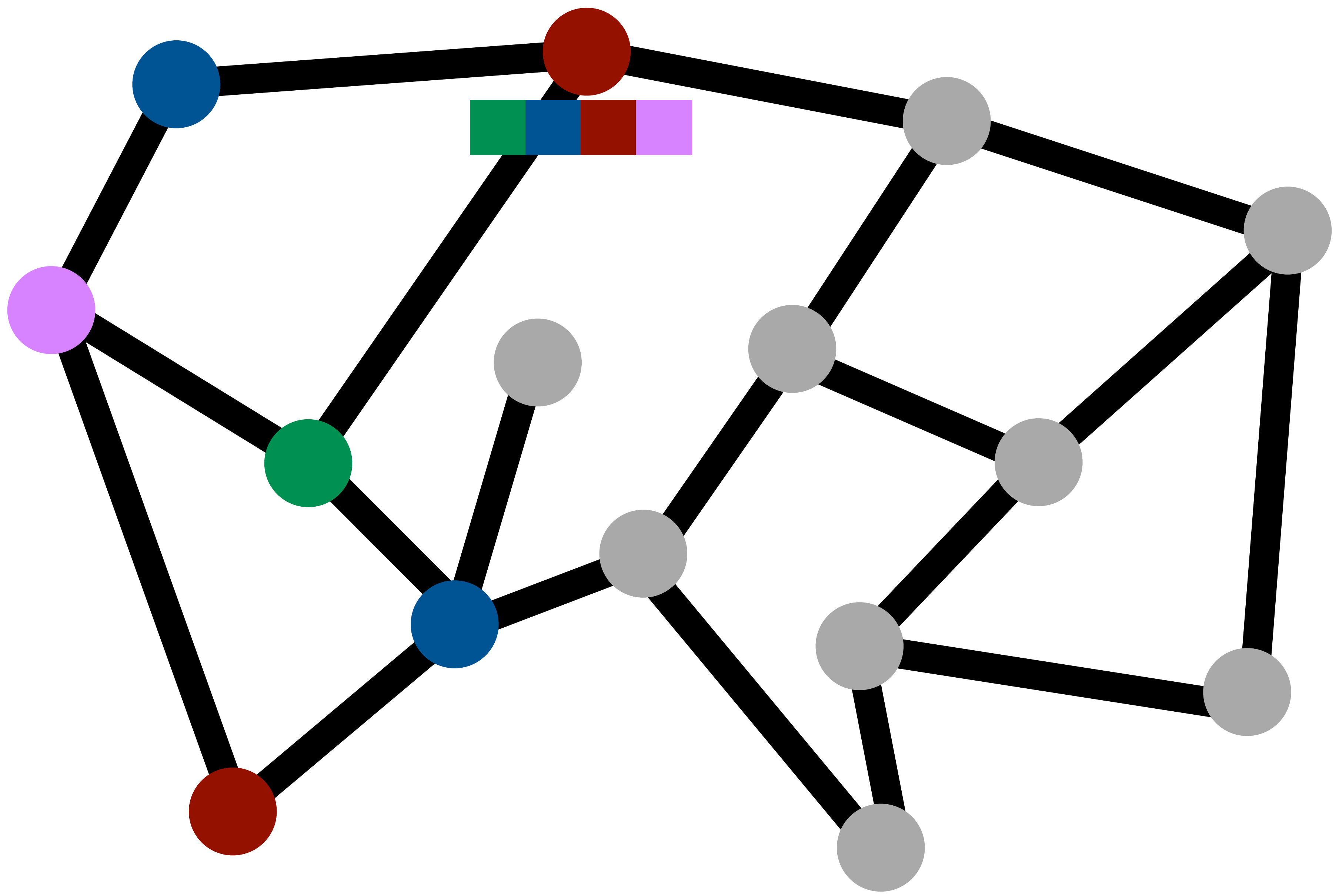
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



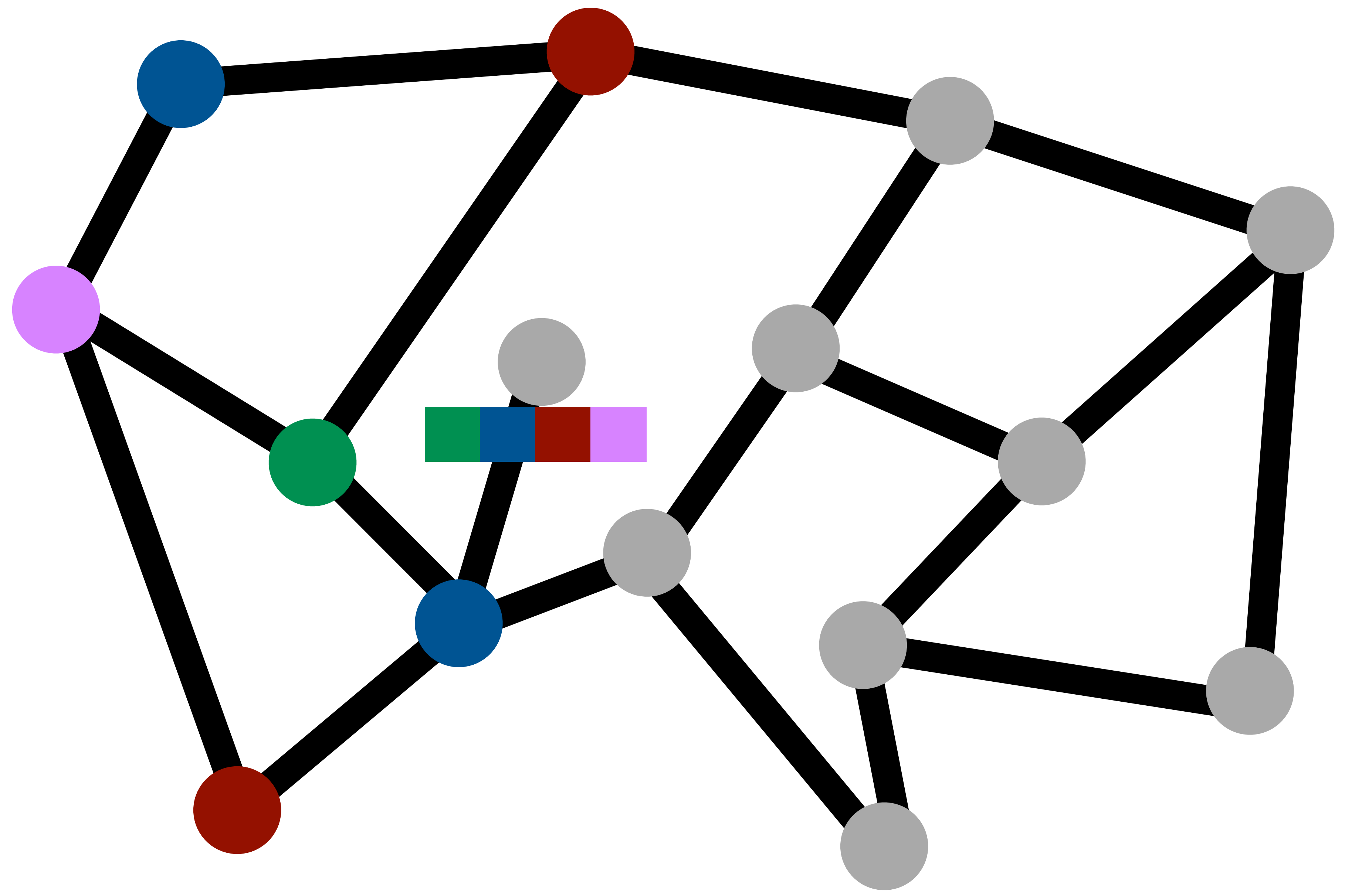
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



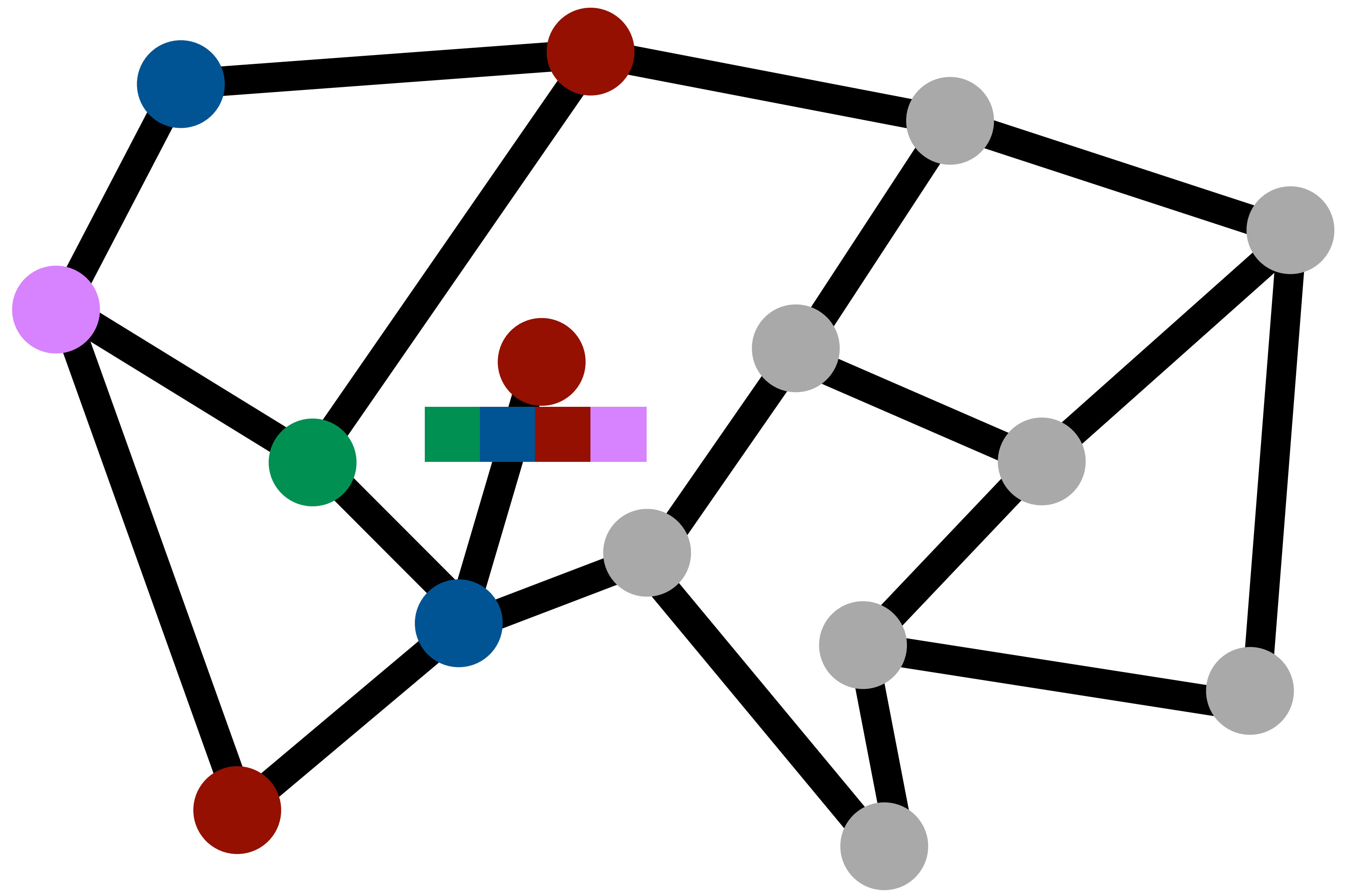
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



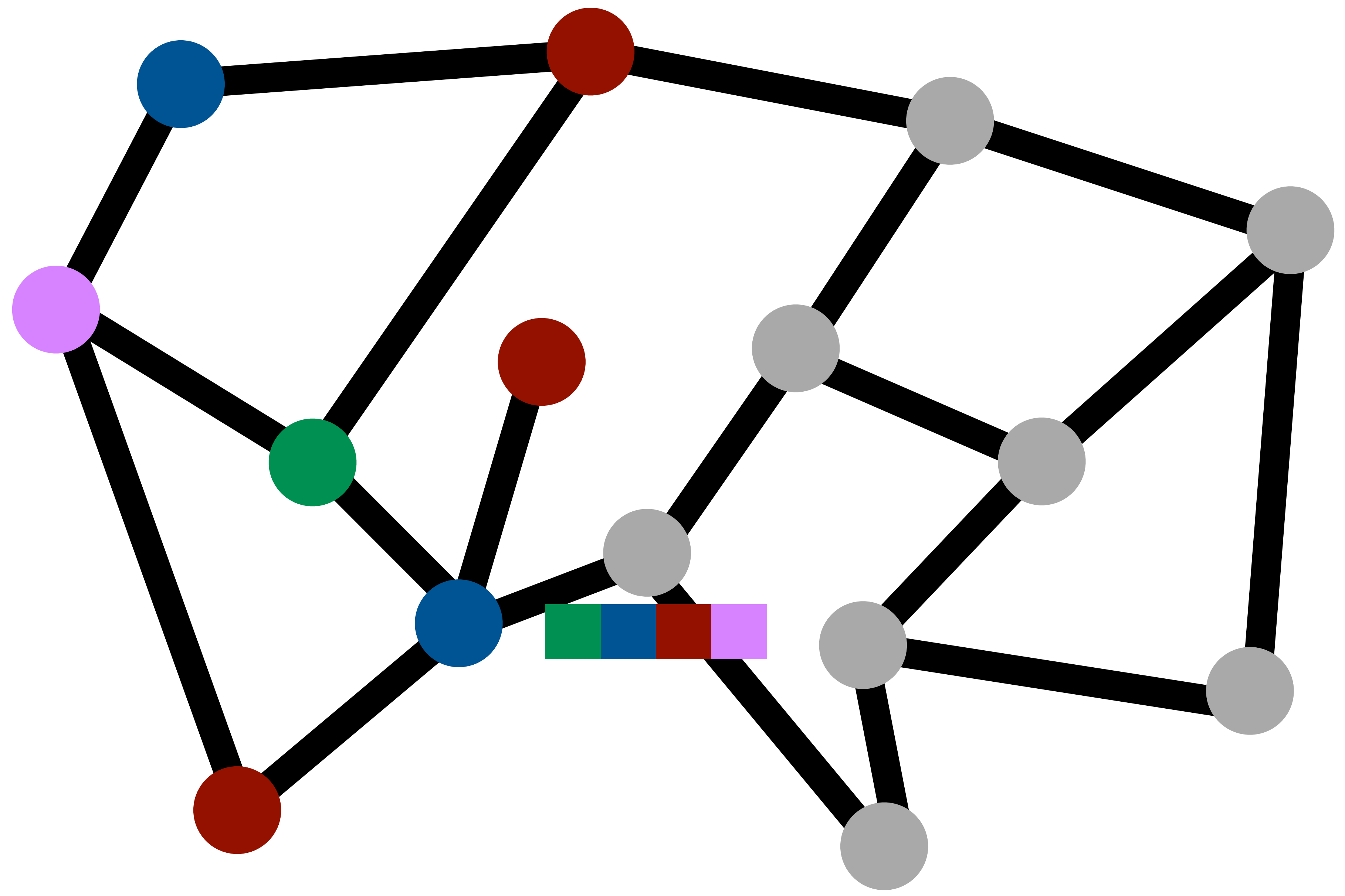
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



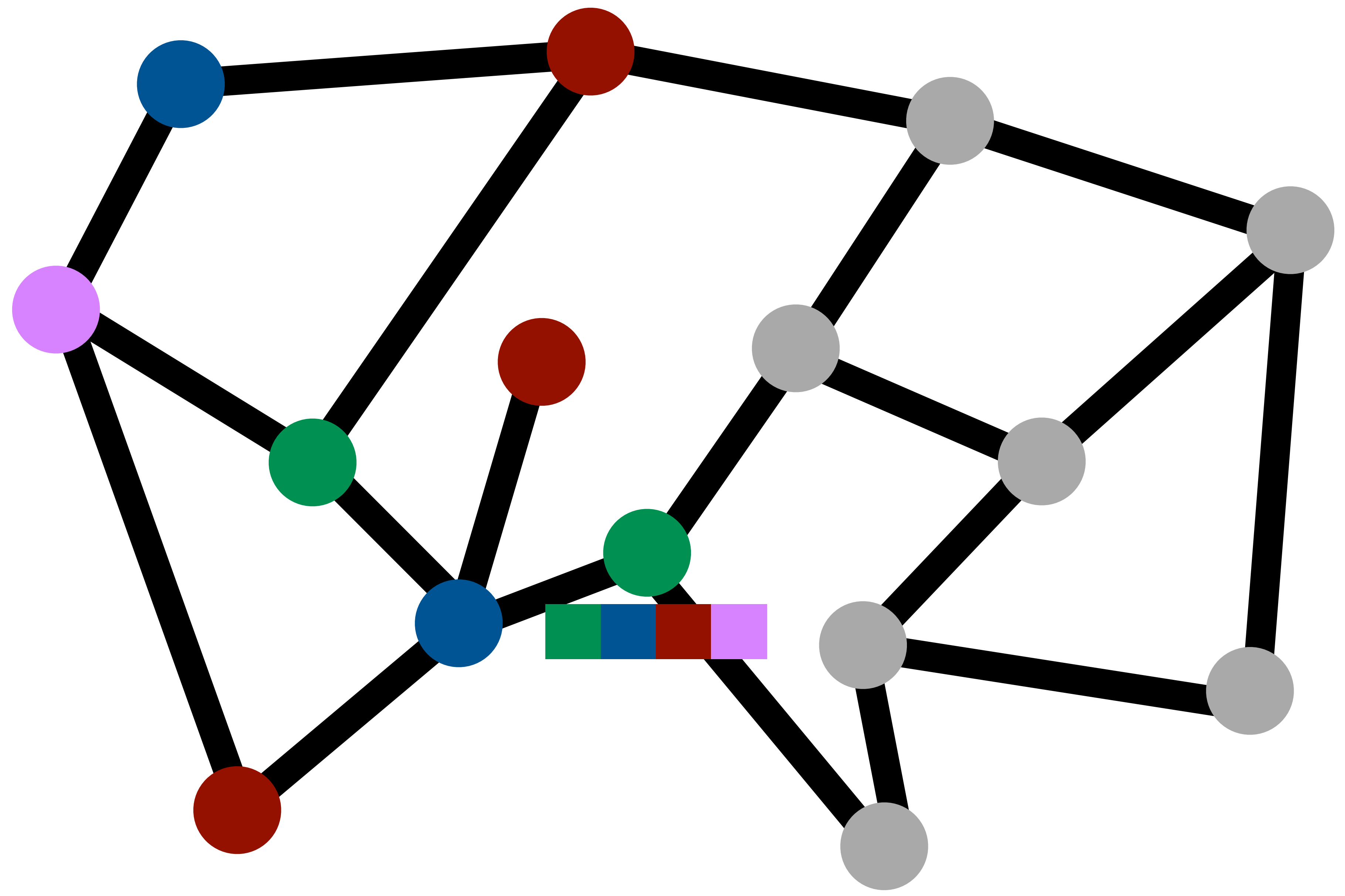
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



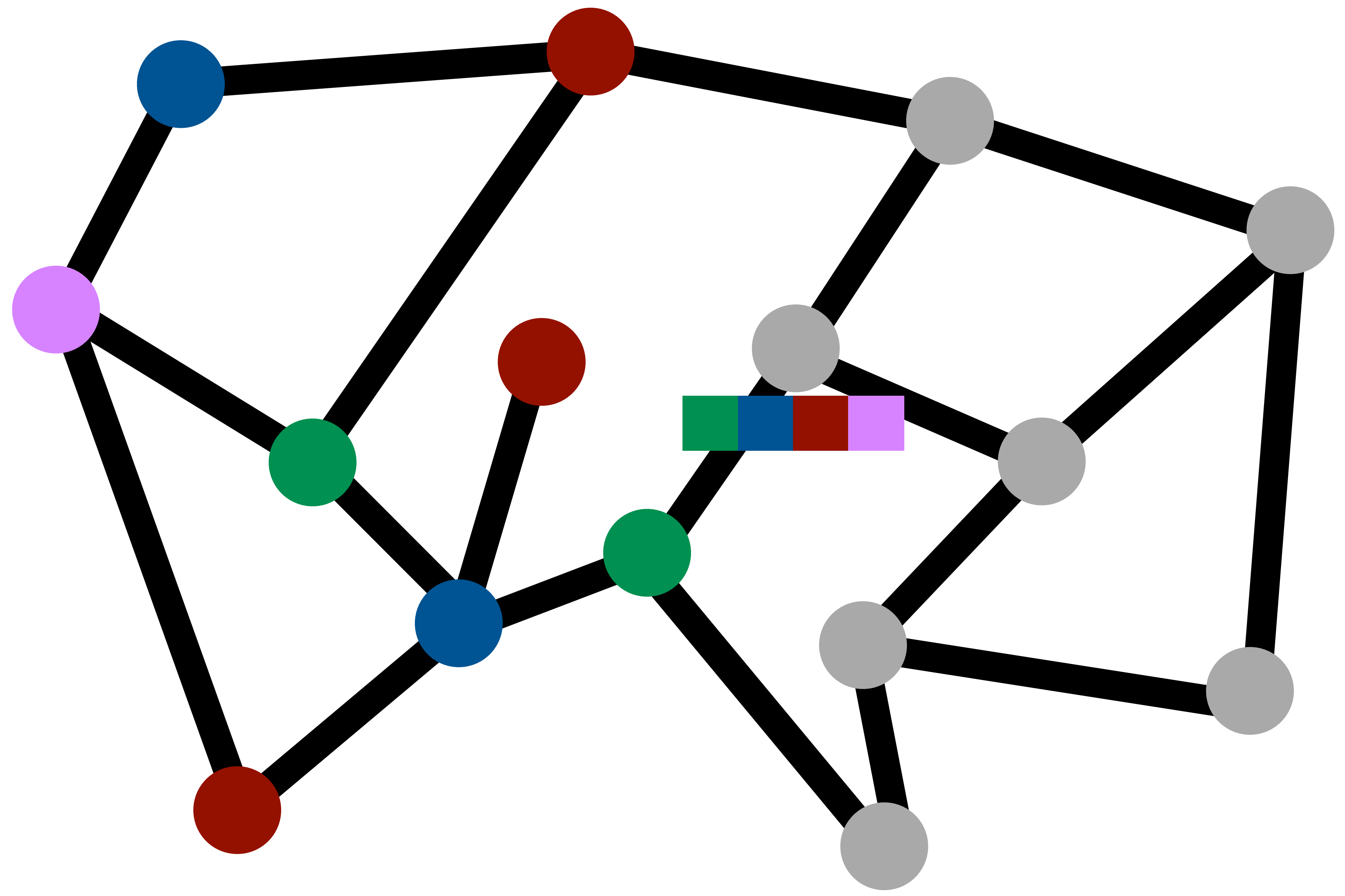
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings

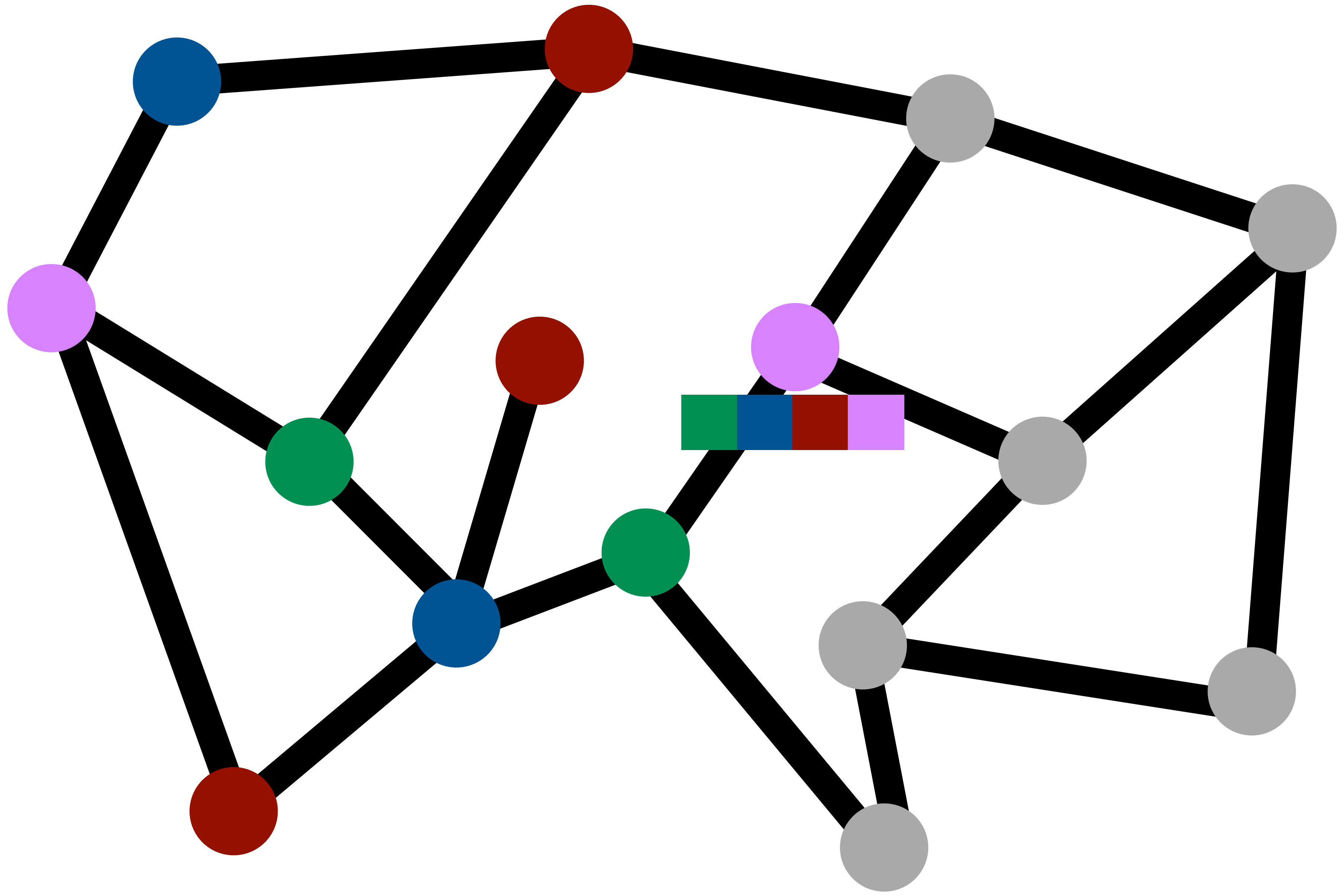


Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Papers Overview

Background: Graph Colorings



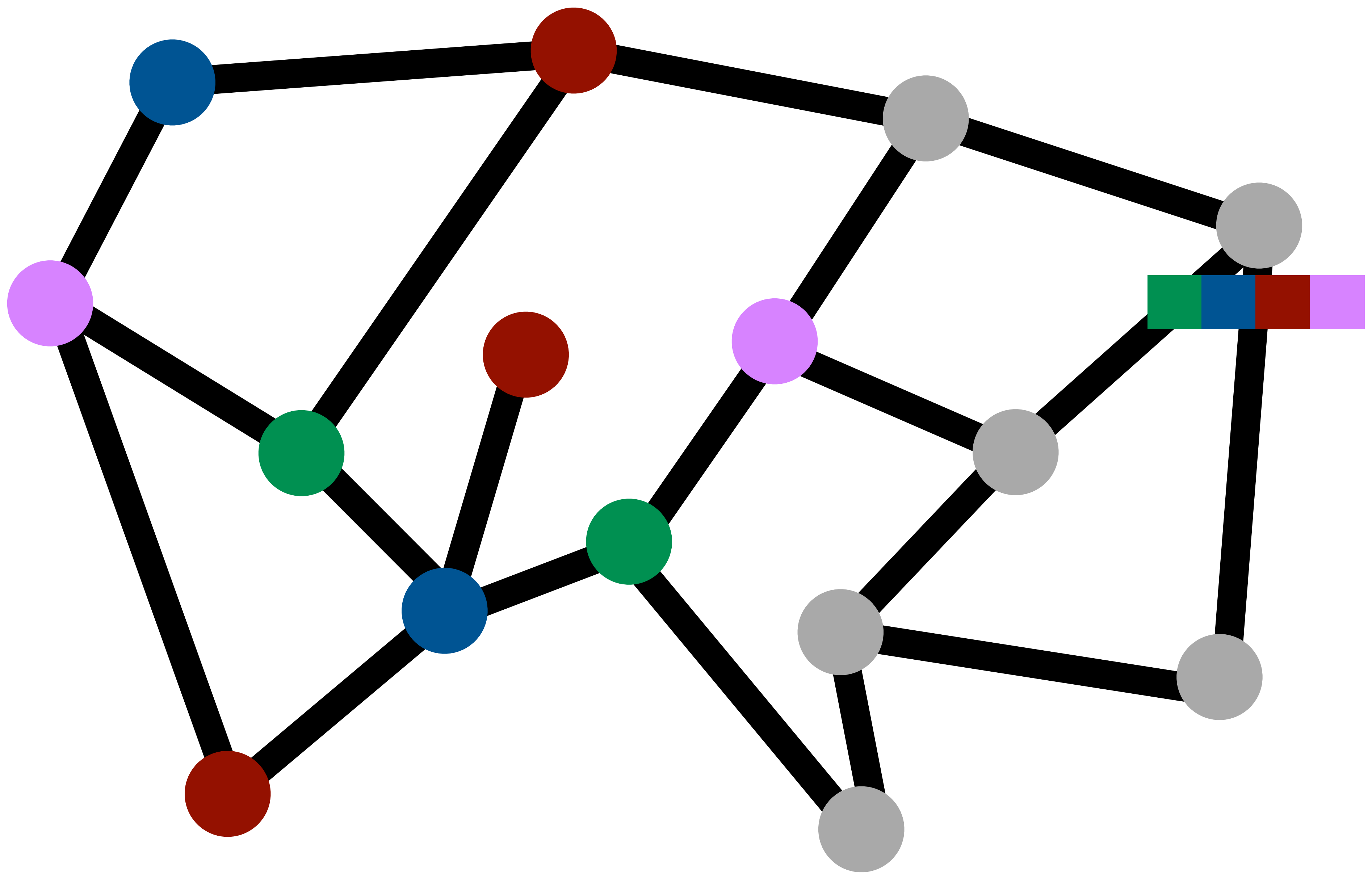
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



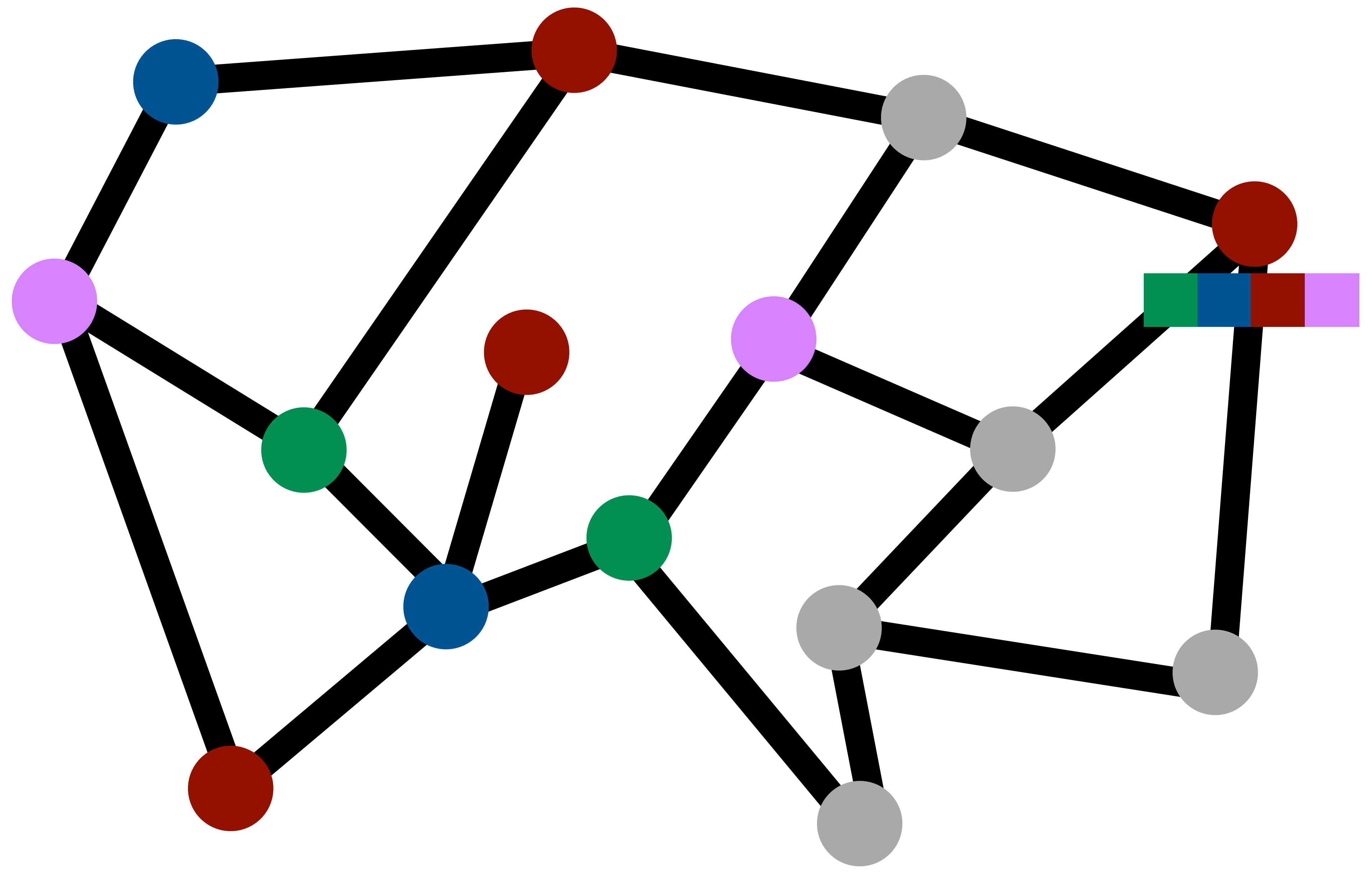
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



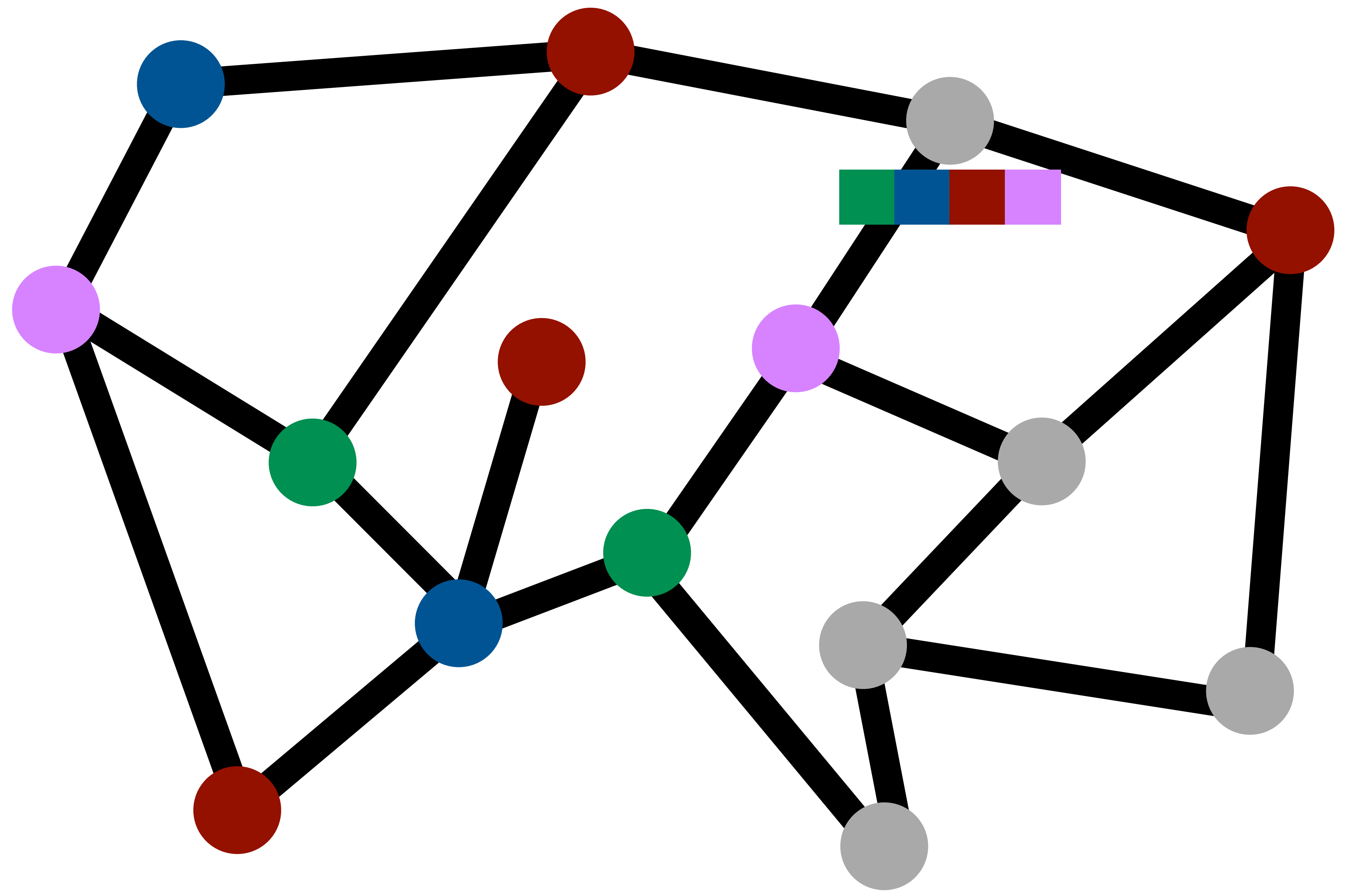
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



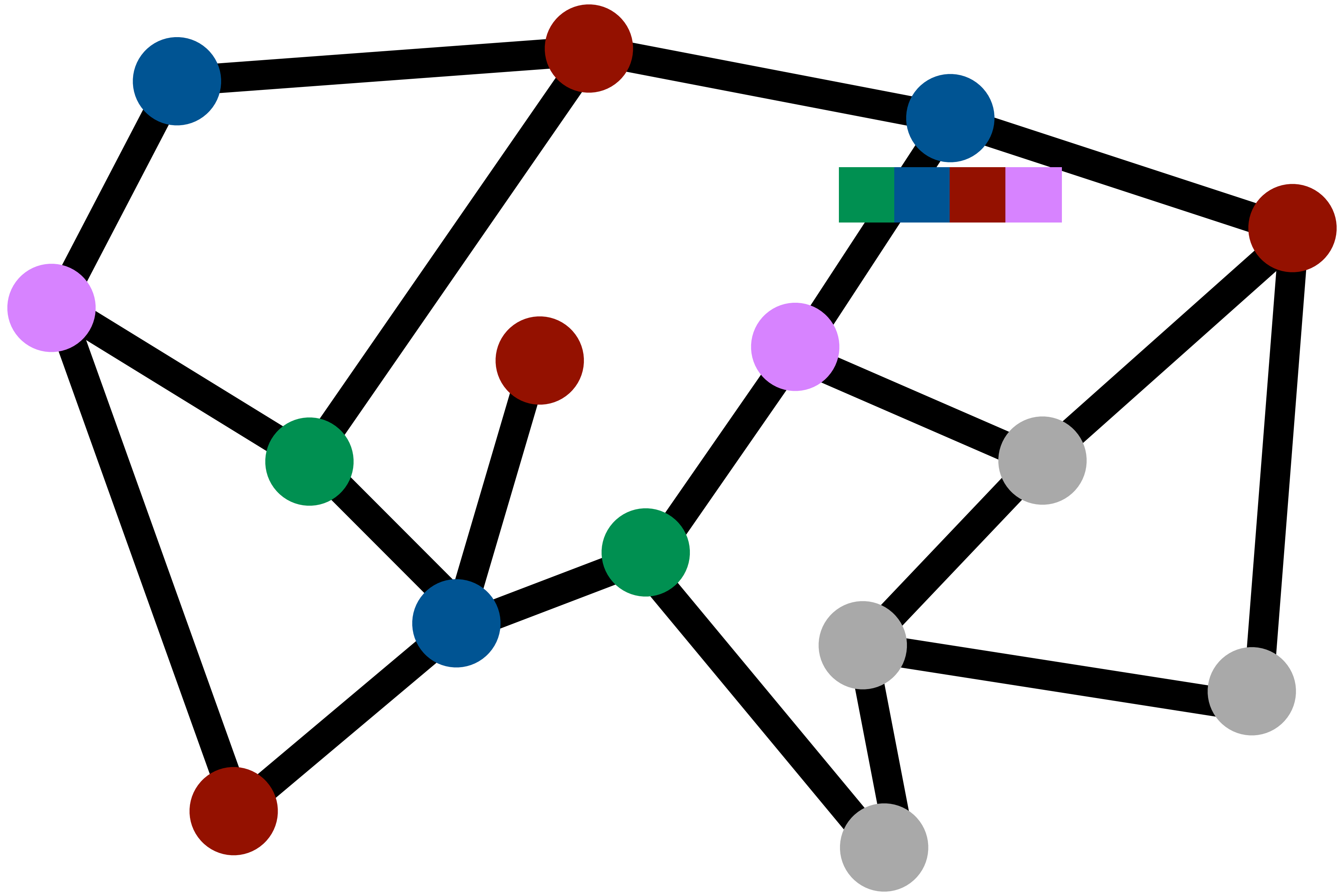
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



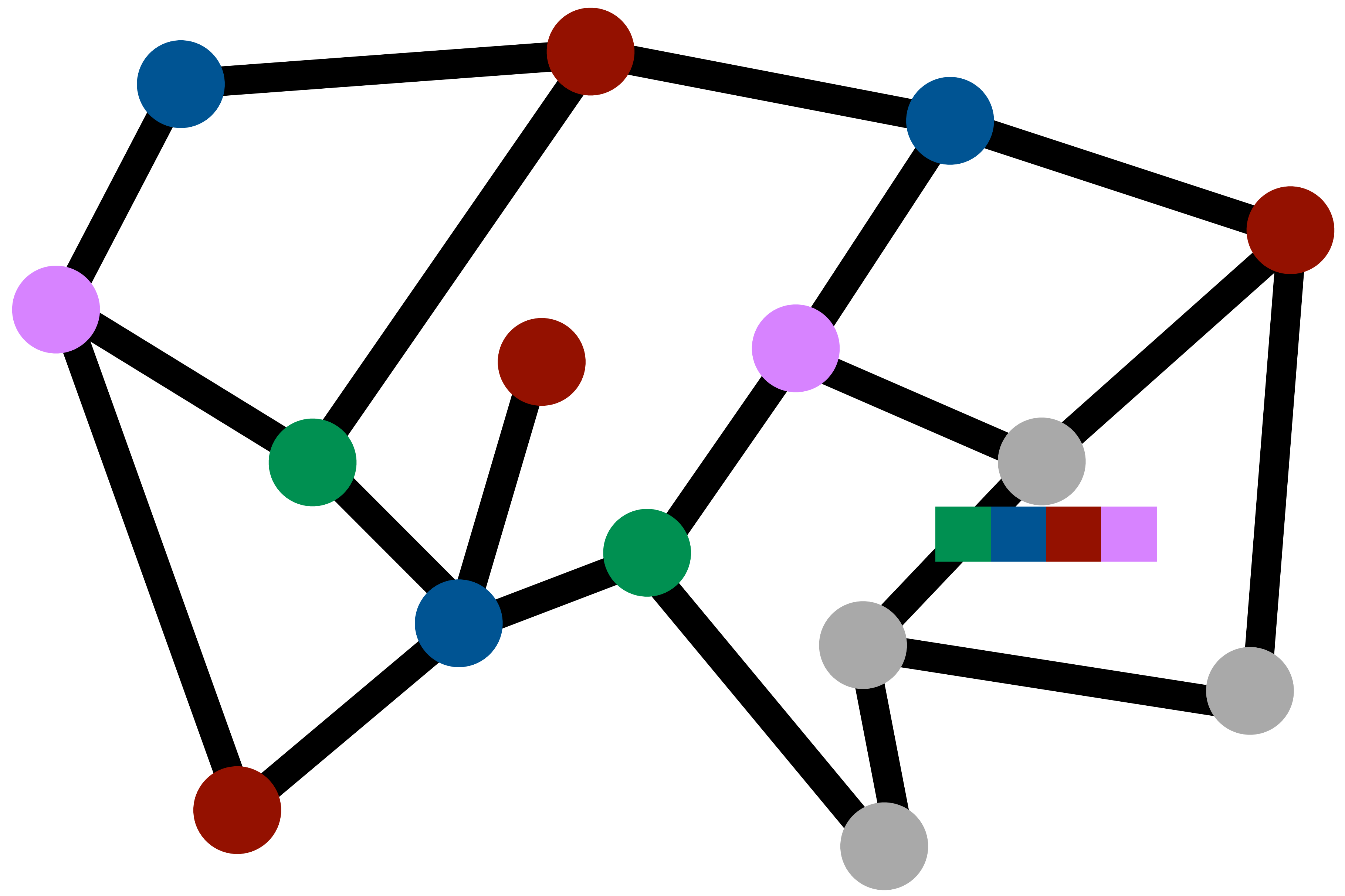
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



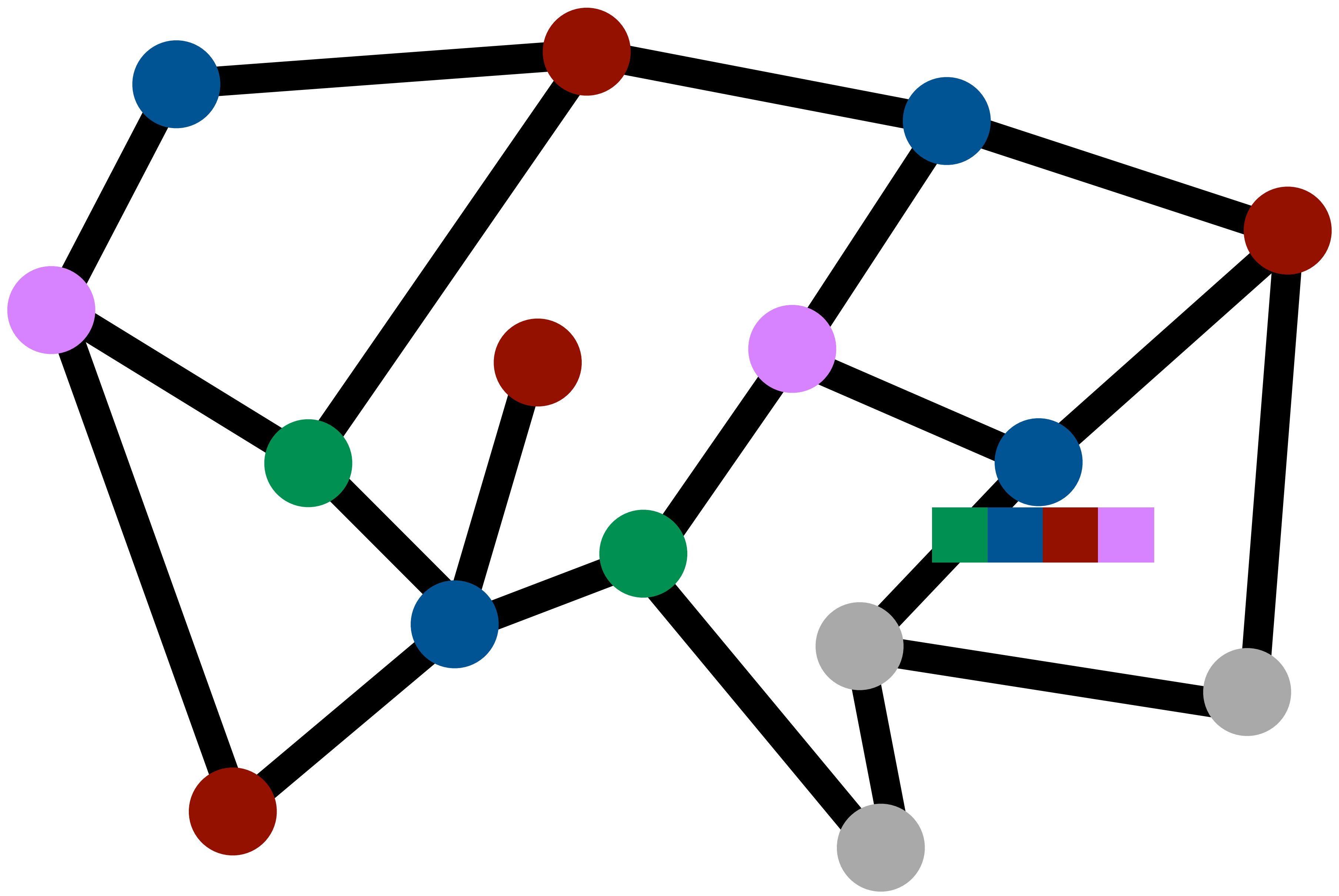
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



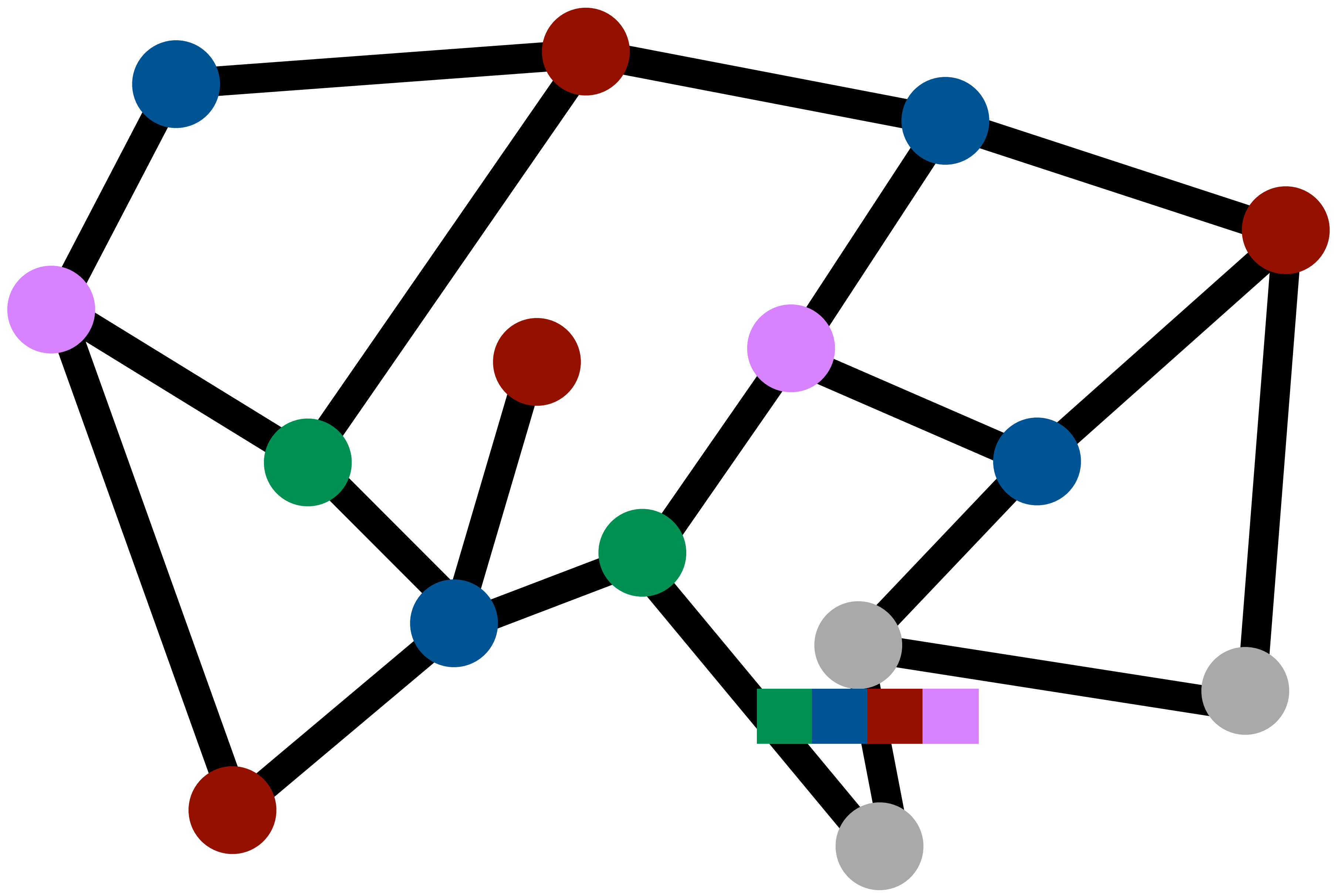
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



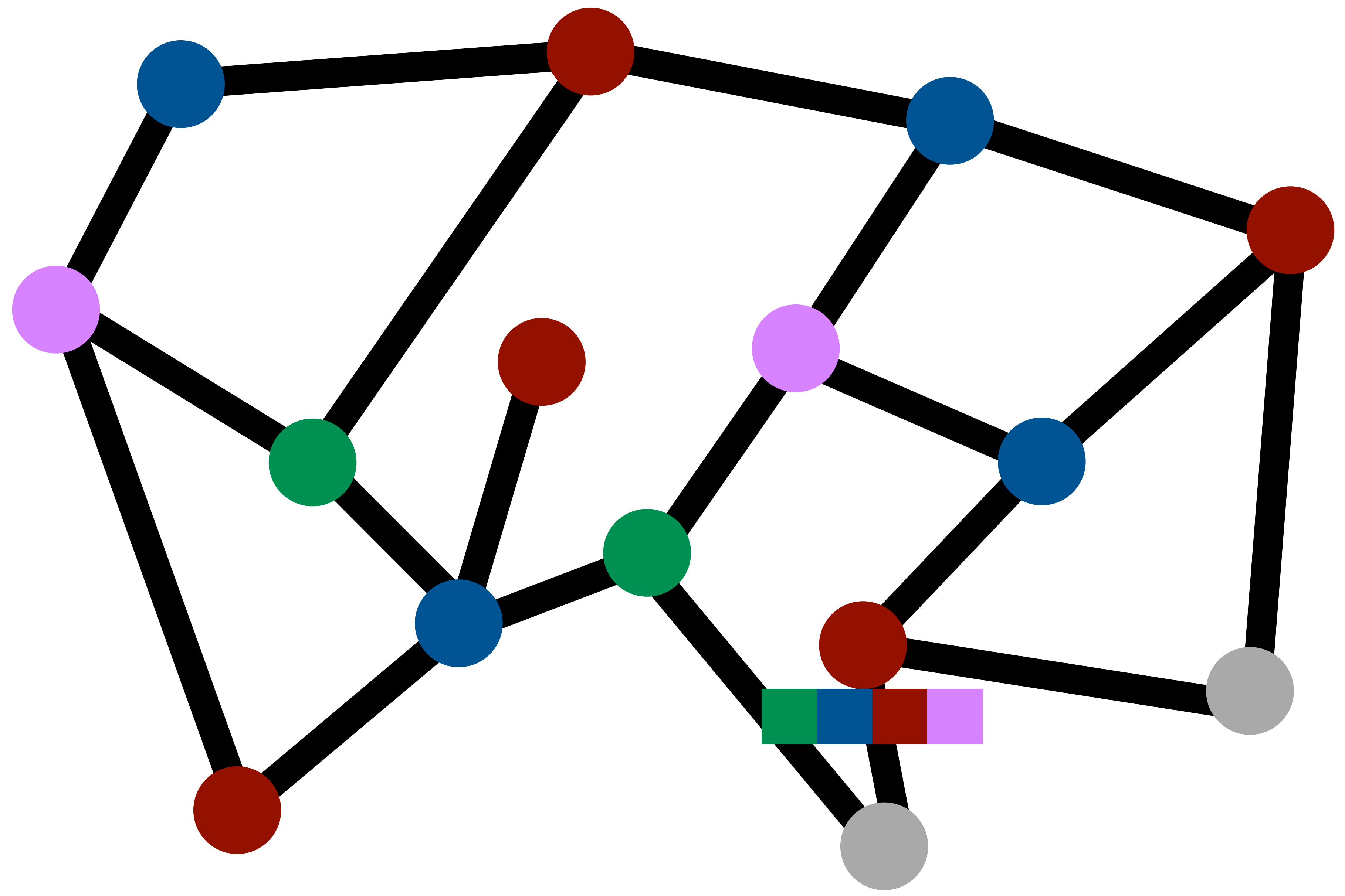
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



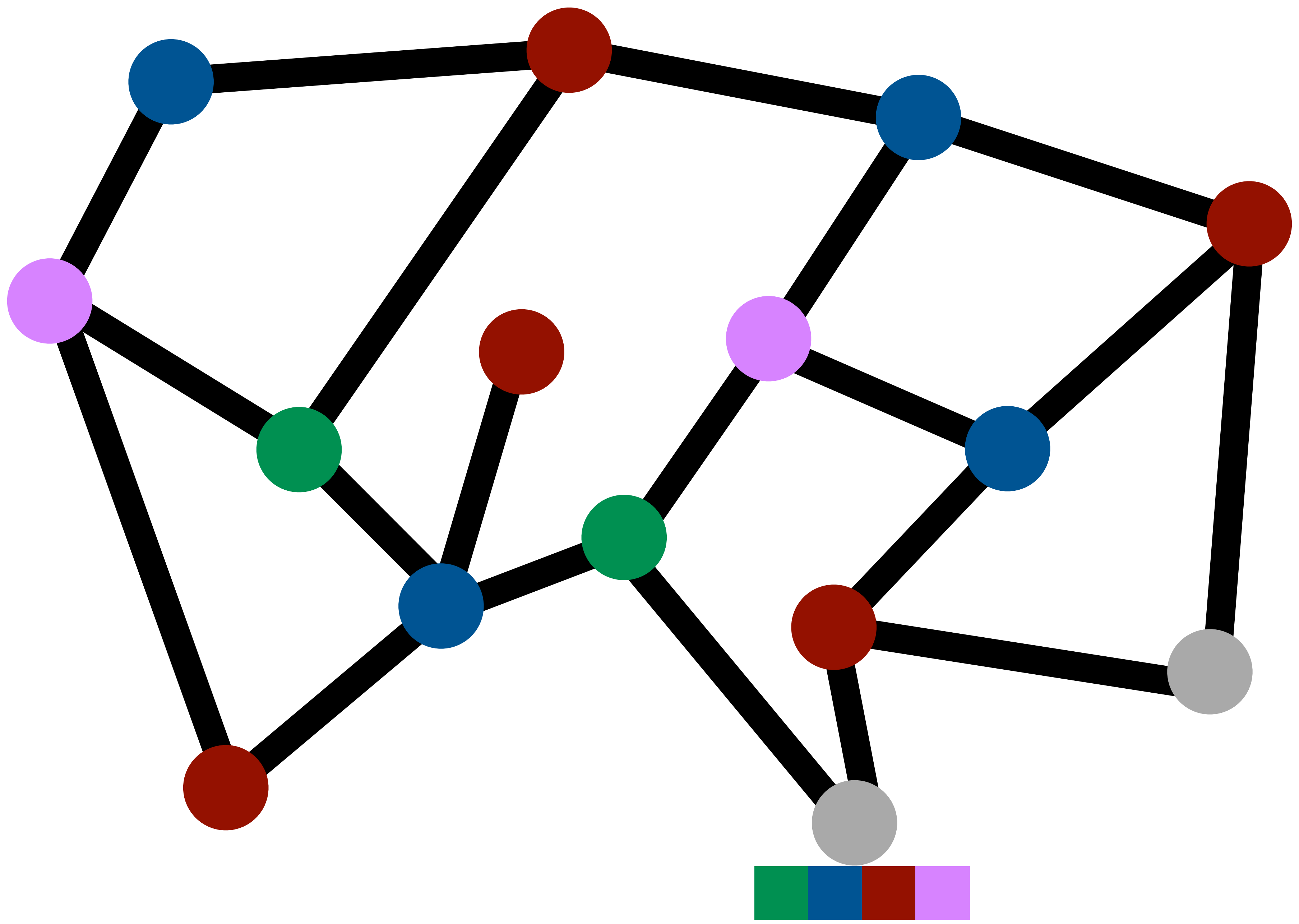
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



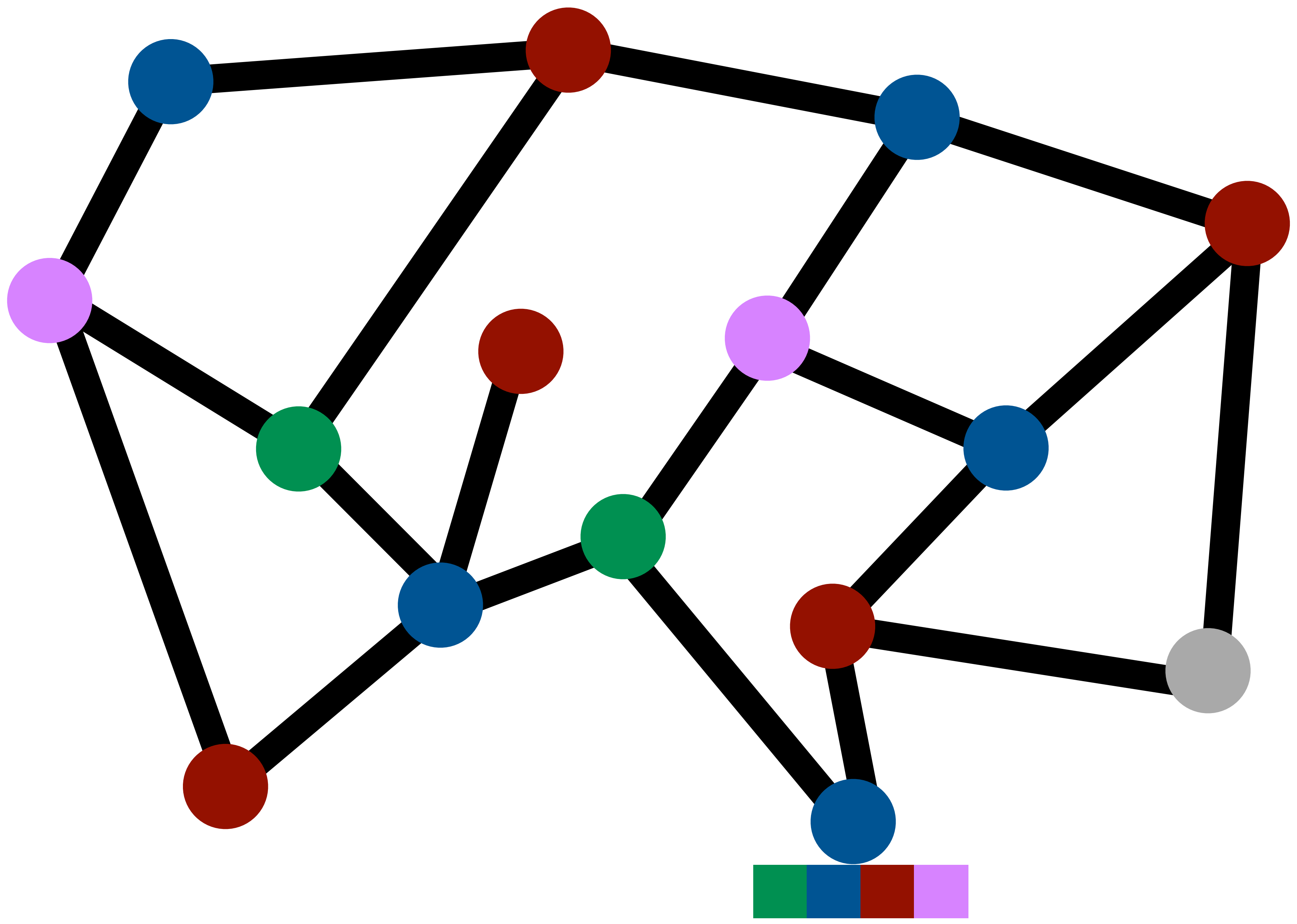
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



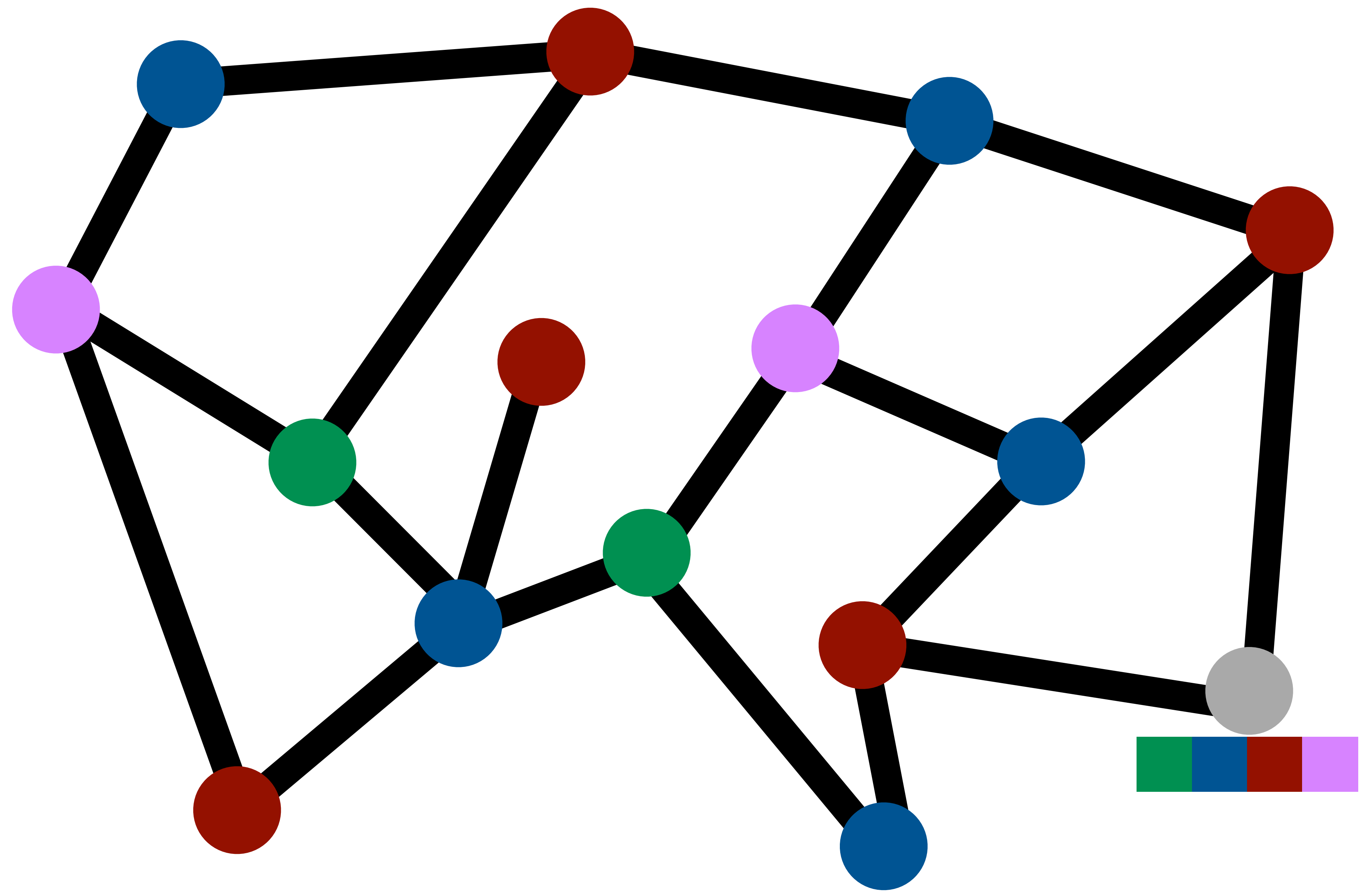
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



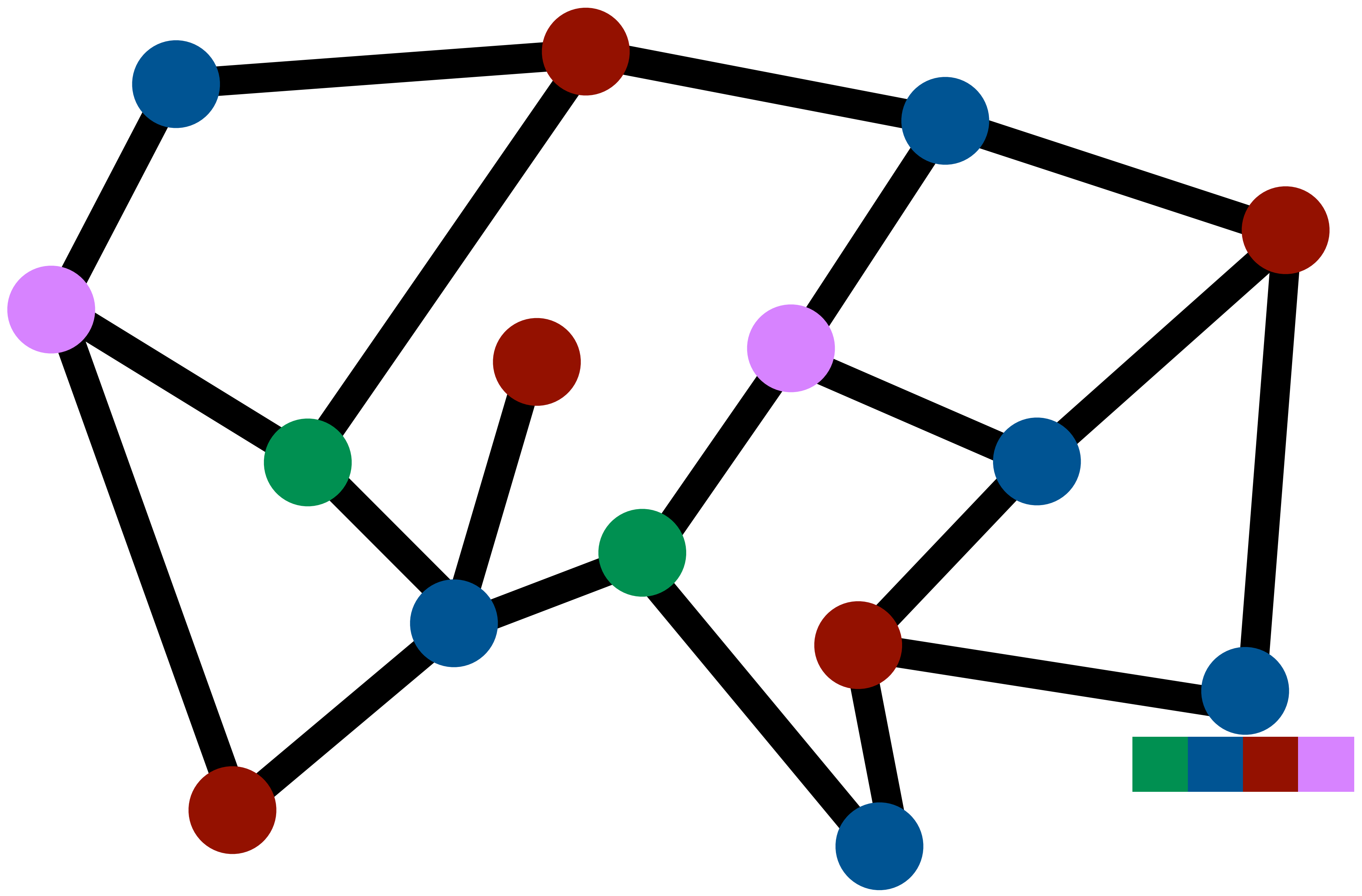
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings



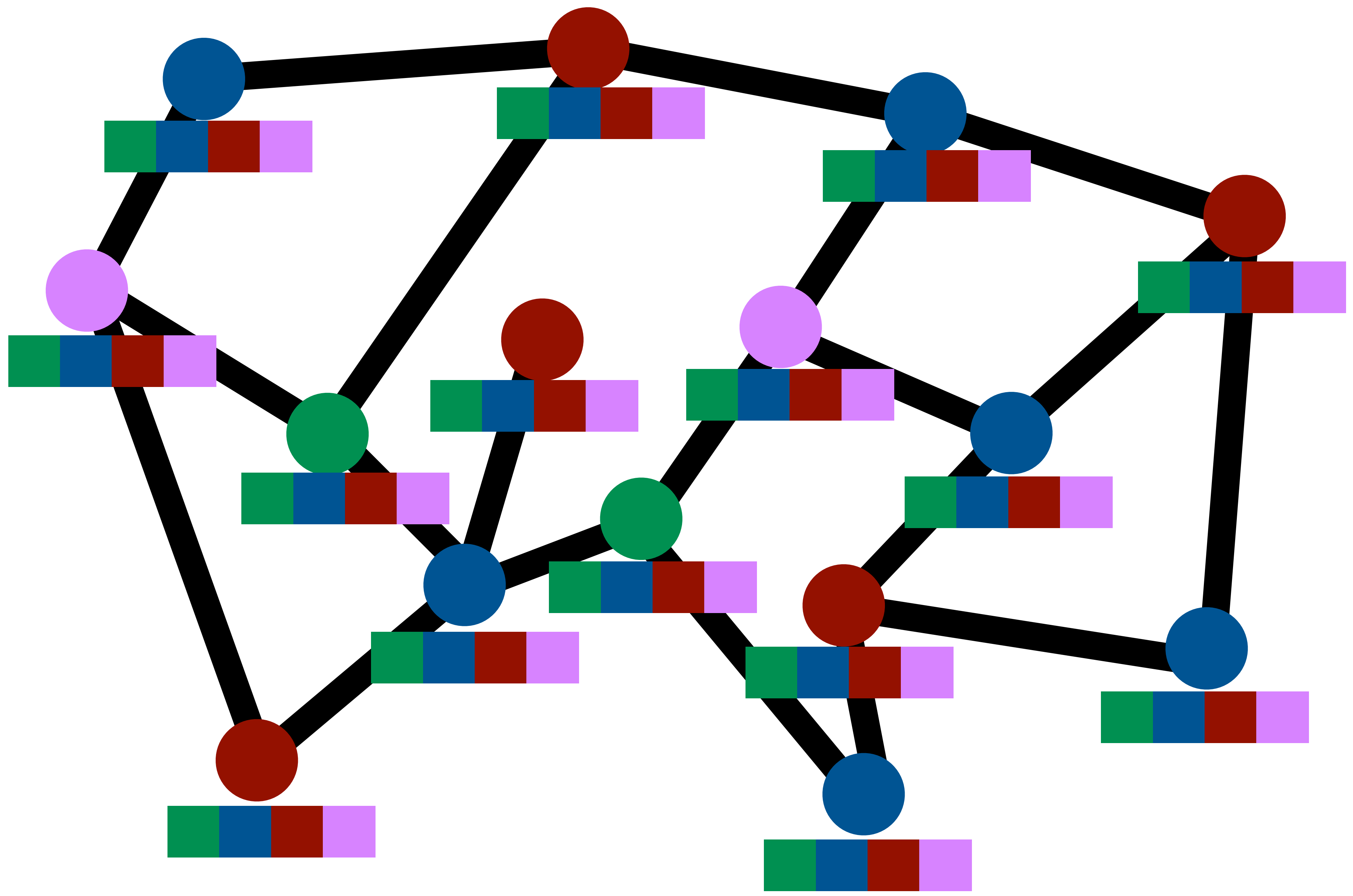
Theorem (folklore): every graph has a $\Delta + 1$ coloring

Proof by greedy algorithm

Δ = max degree

Papers Overview

Background: Graph Colorings

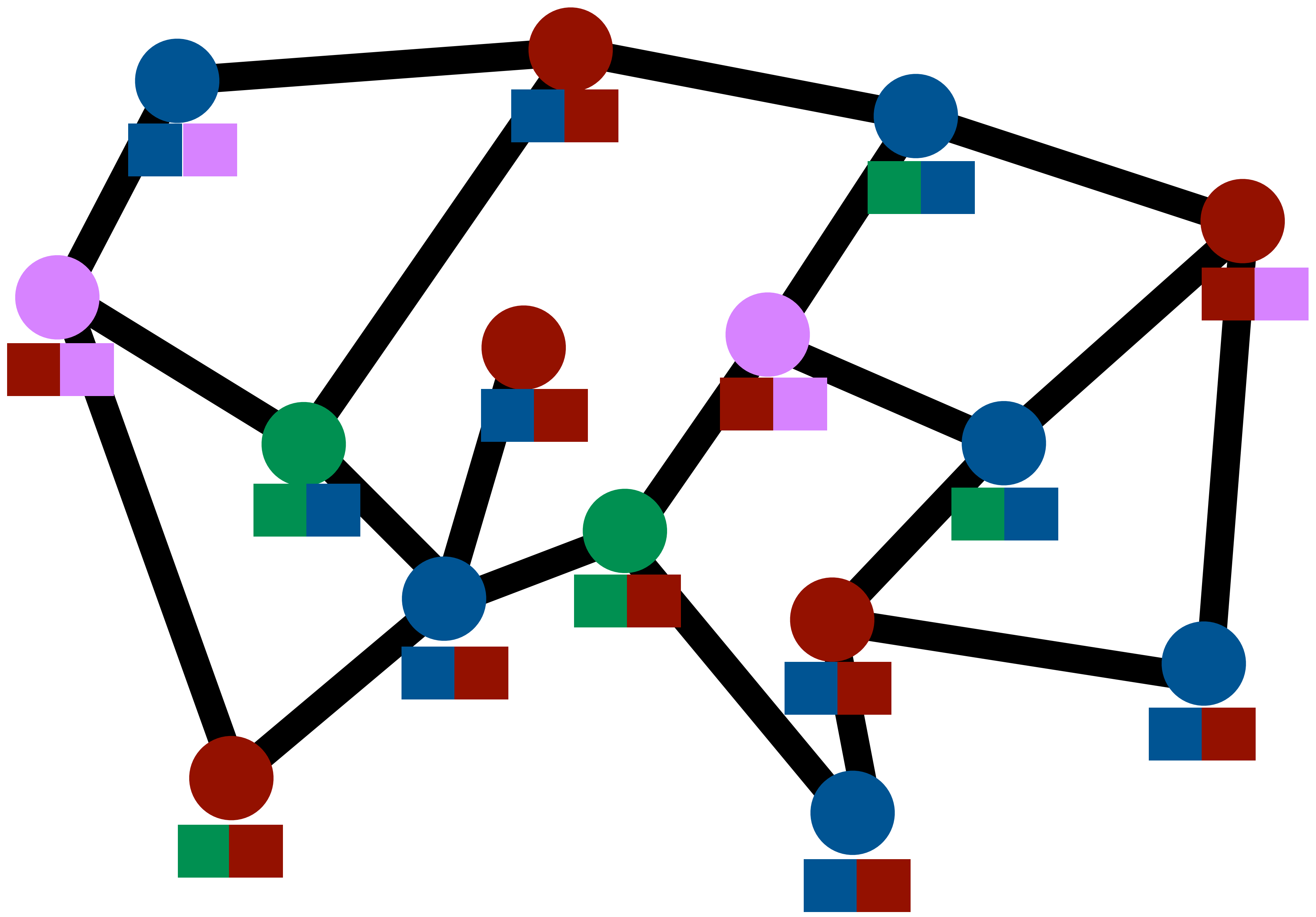


Theorem (folklore): can color a graph if every vertex has a "palette" of $\Delta + 1$ colors

Δ = max degree

Papers Overview

Paper 10: Palette Sparsification

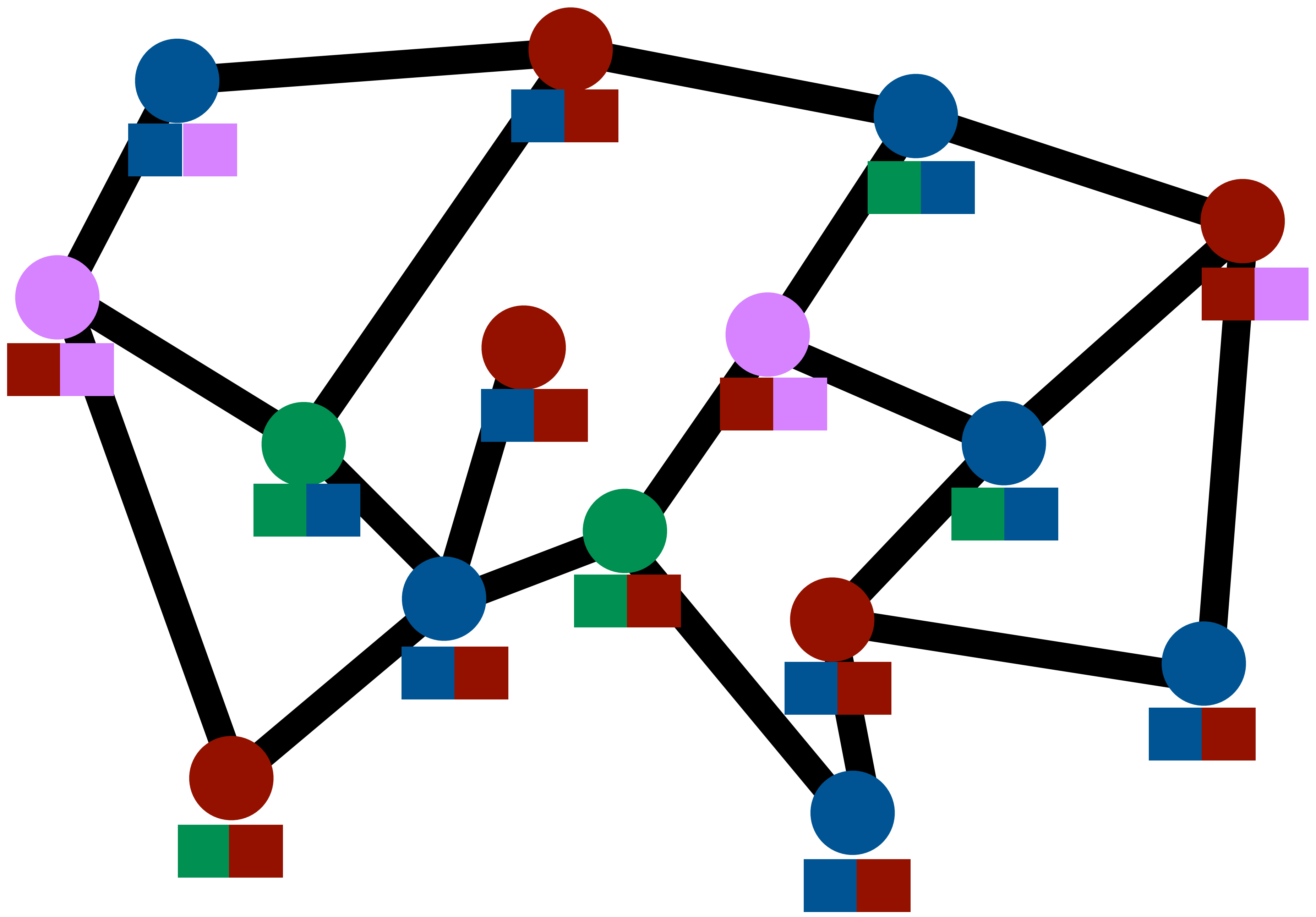


Theorem: can color a graph if each vertex samples a palette of size $\Omega(\log n)$ from $\Delta + 1$ colors

Δ = max degree

Papers Overview

Paper 10: Palette Sparsification

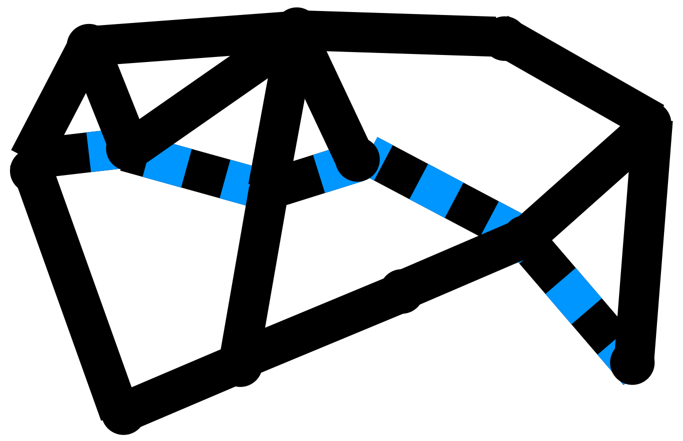


Theorem(informal): can efficiently color a graph in *many* models of computation

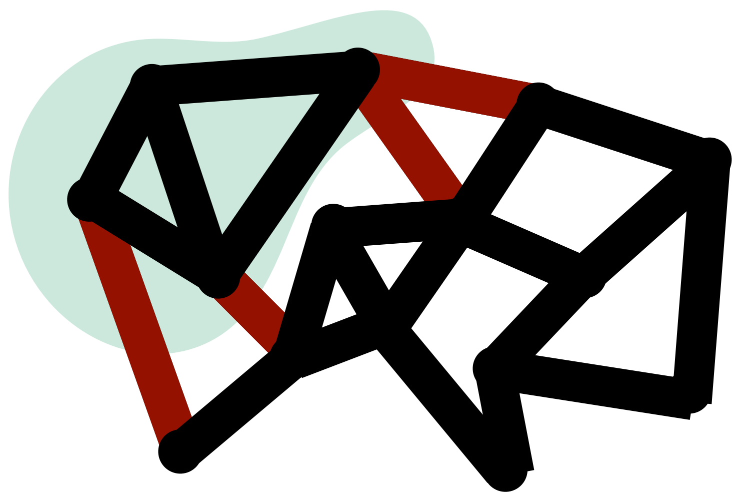
Papers Overview

Sparsification of Five Graph-Theoretic Objects

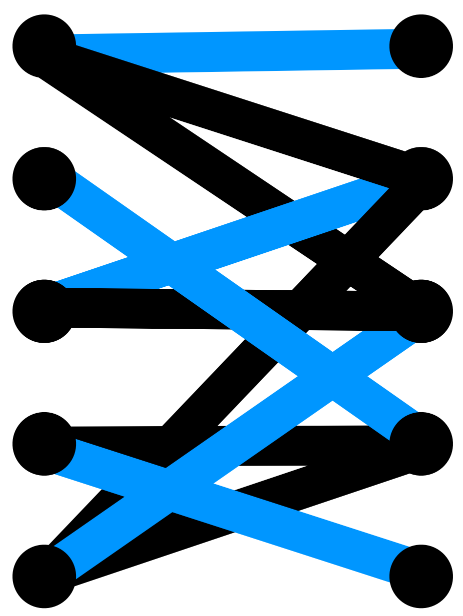
Distances



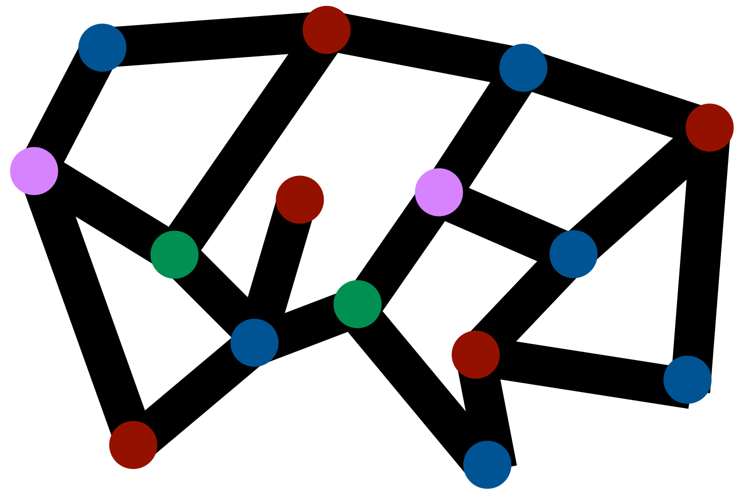
Cuts/Flows



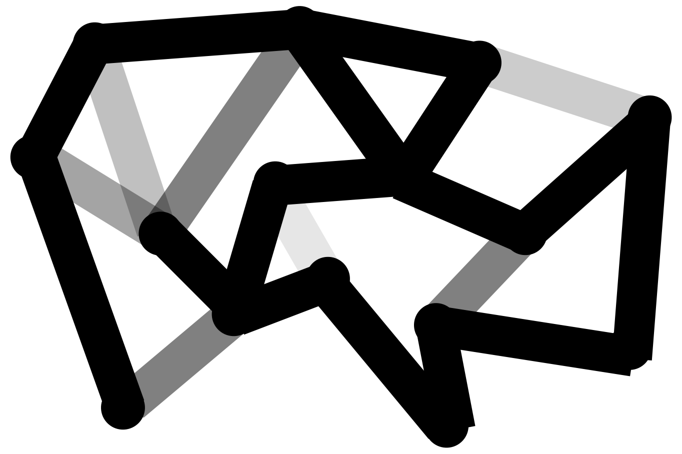
Matchings



Colorings



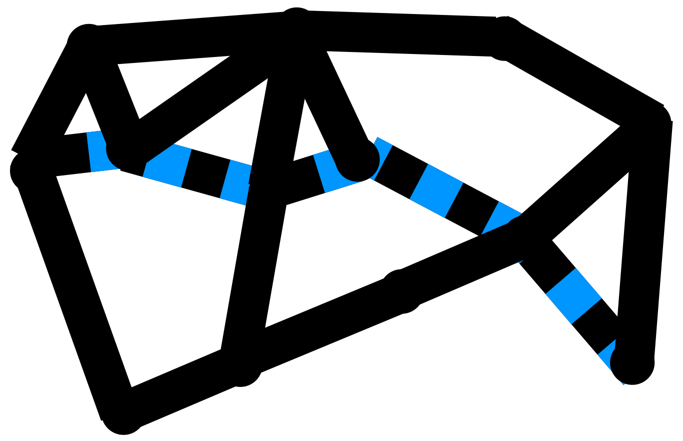
Fractional Opts



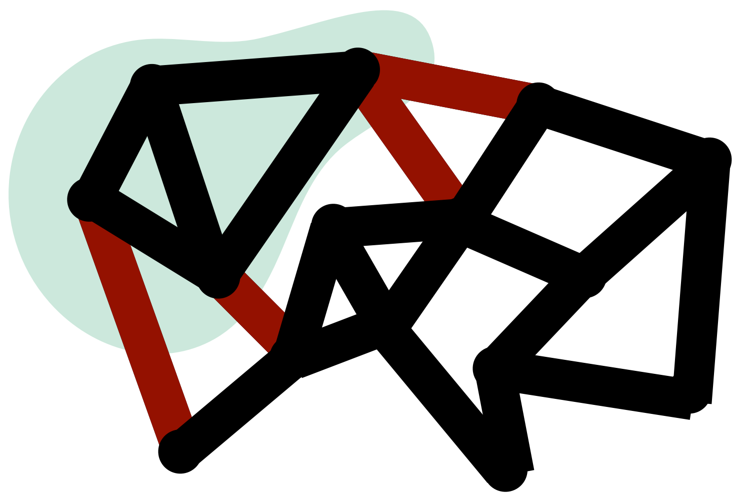
Papers Overview

Sparsification of Five Graph-Theoretic Objects

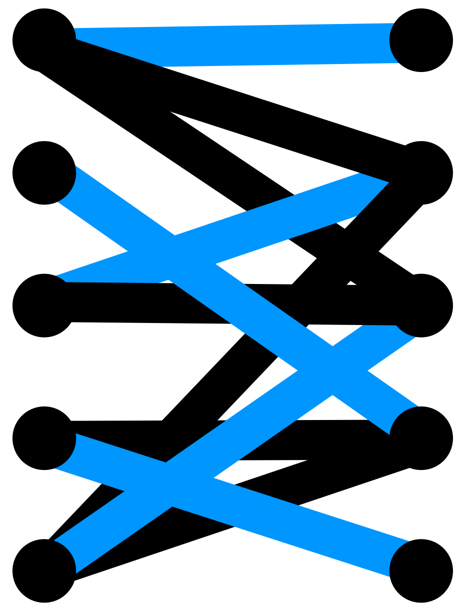
Distances



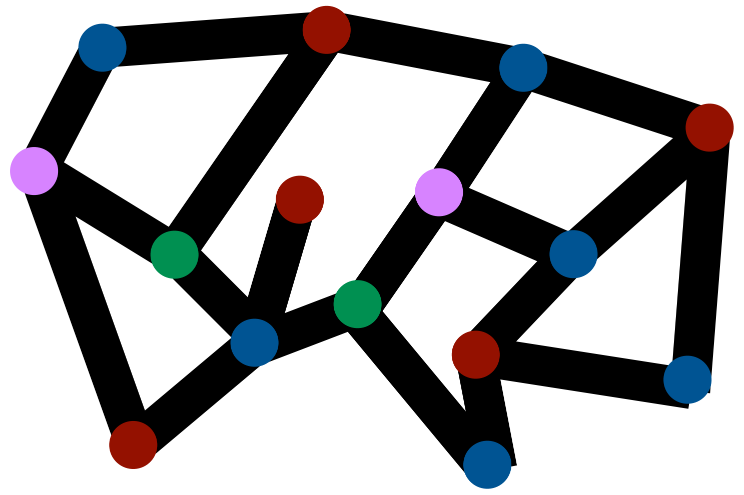
Cuts/Flows



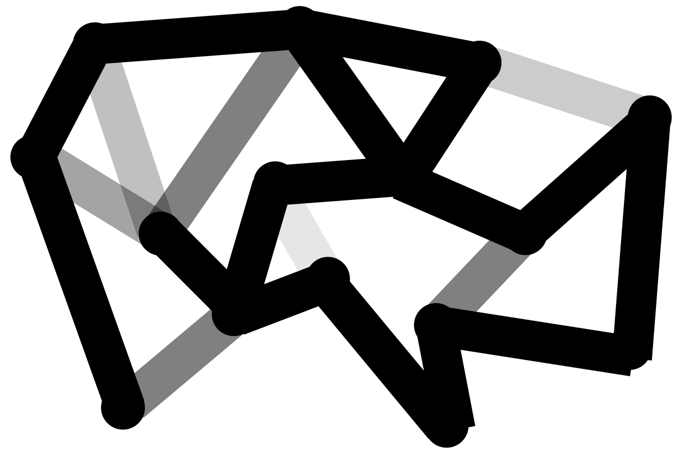
Matchings



Colorings

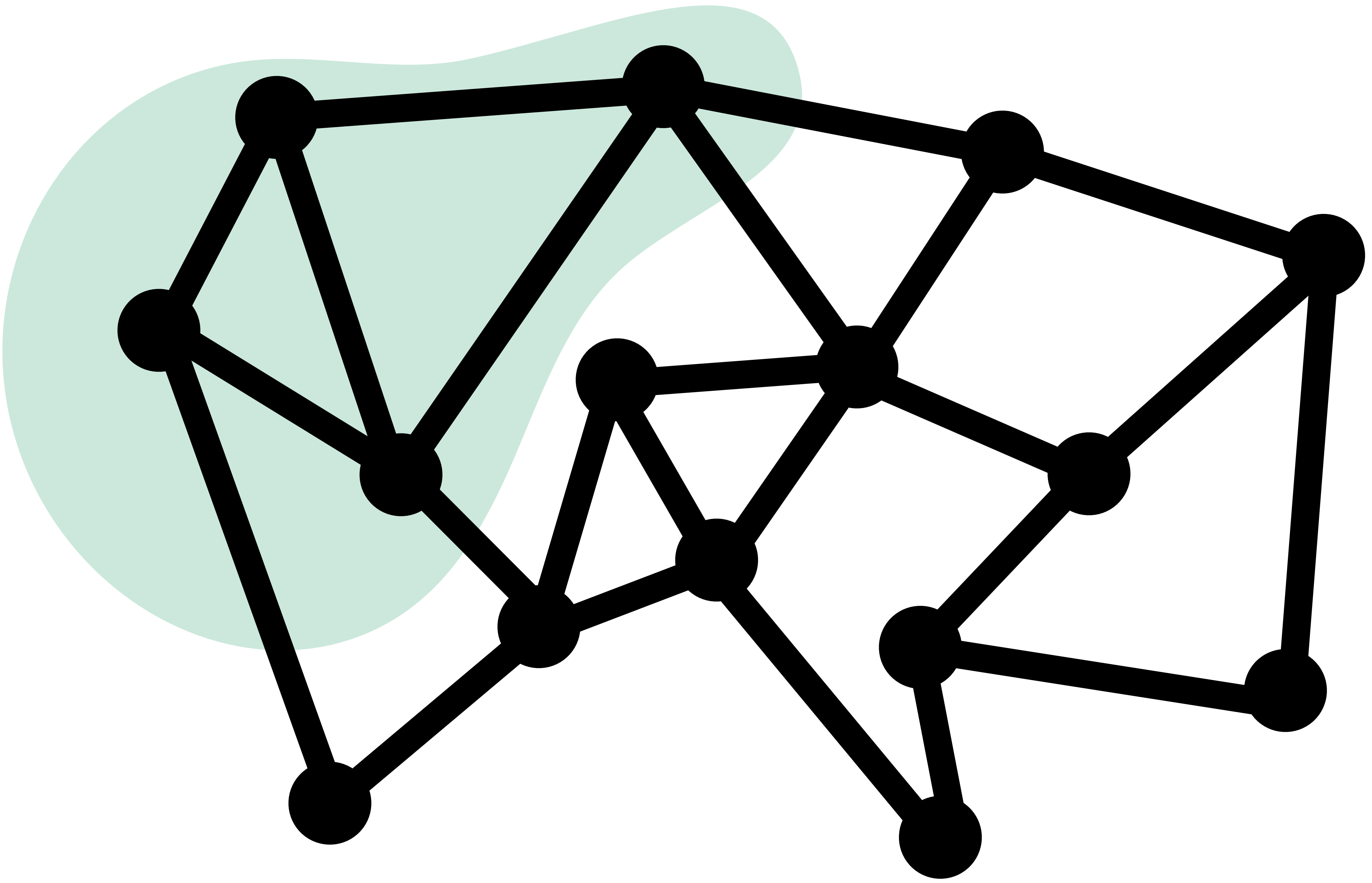


Fractional Opts



Papers Overview

Background: Survivable Network Design



graph $G = (V, E, w)$

E.g. $|\delta(S)| \geq 1$

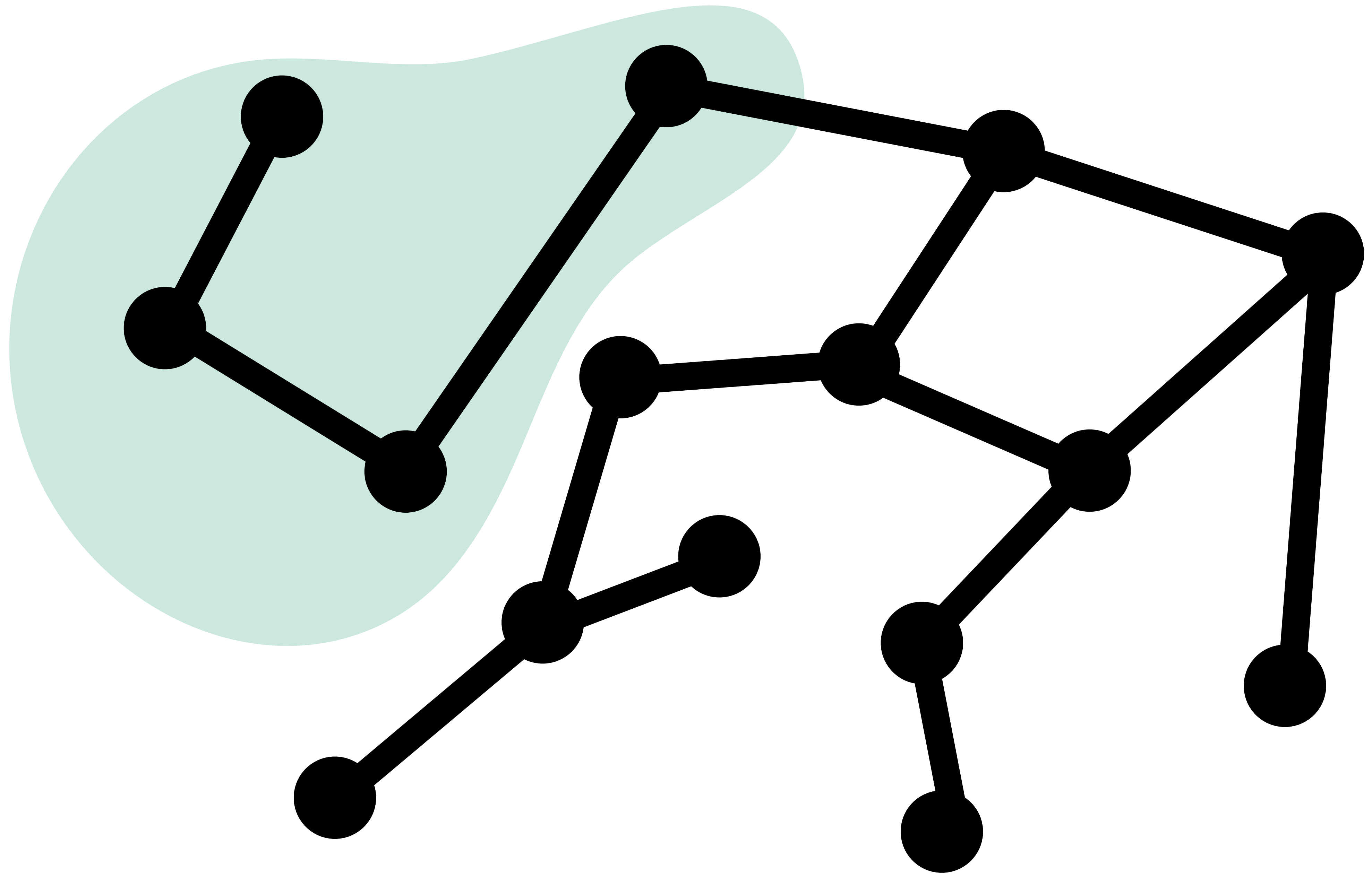
$\forall S \subset V$

MST $\in P$

Goal: efficiently find subset of min-weight subgraph satisfying *cut constraints*

Papers Overview

Background: Survivable Network Design



graph $G = (V, E, w)$

E.g. $|\delta(S)| \geq 1$

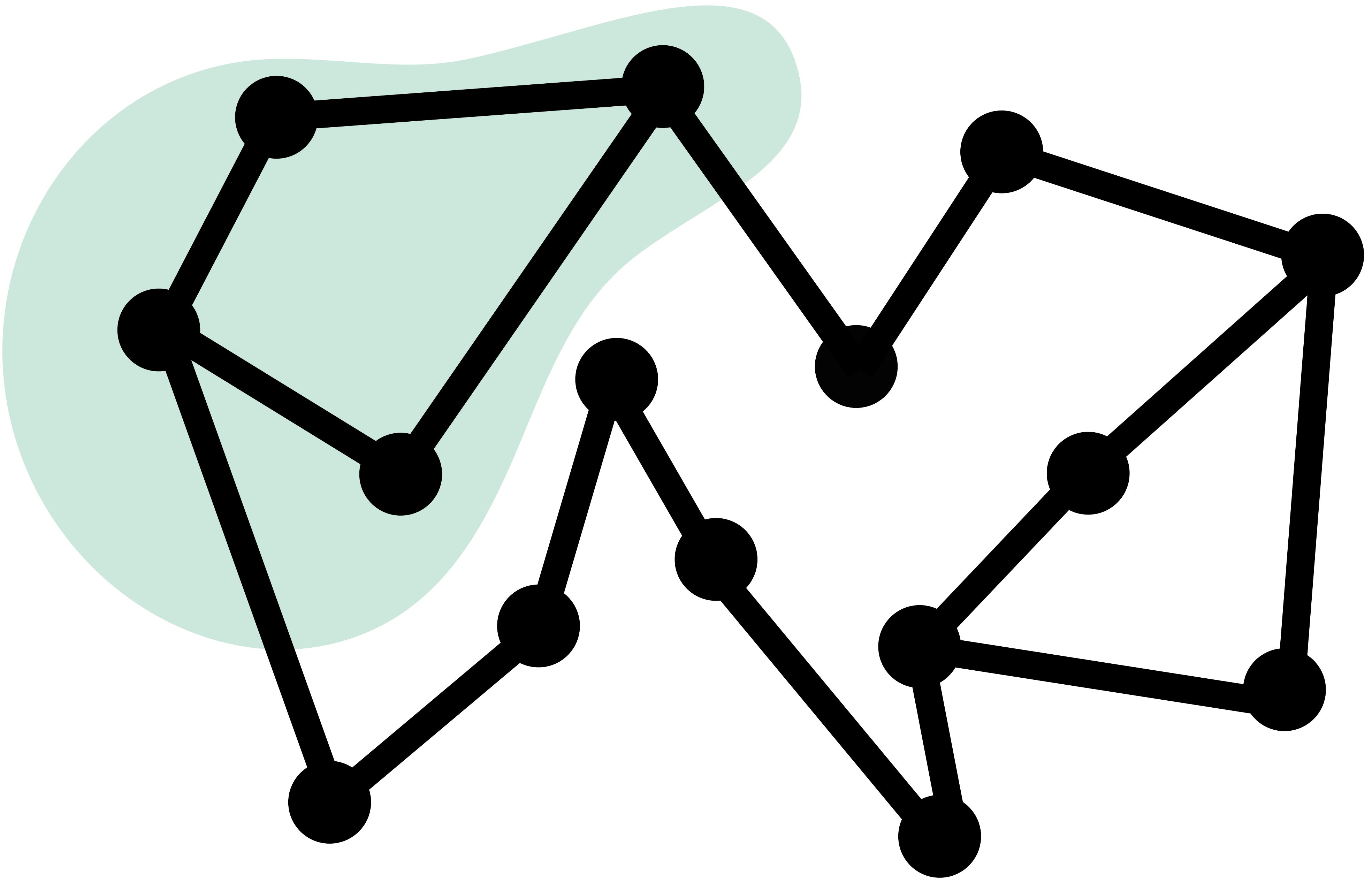
$\forall S \subset V$

MST $\in P$

Goal: efficiently find subset of min-weight subgraph satisfying *cut constraints*

Papers Overview

Background: Survivable Network Design



graph $G = (V, E, w)$

E.g. $|\delta(S)| \geq 2$

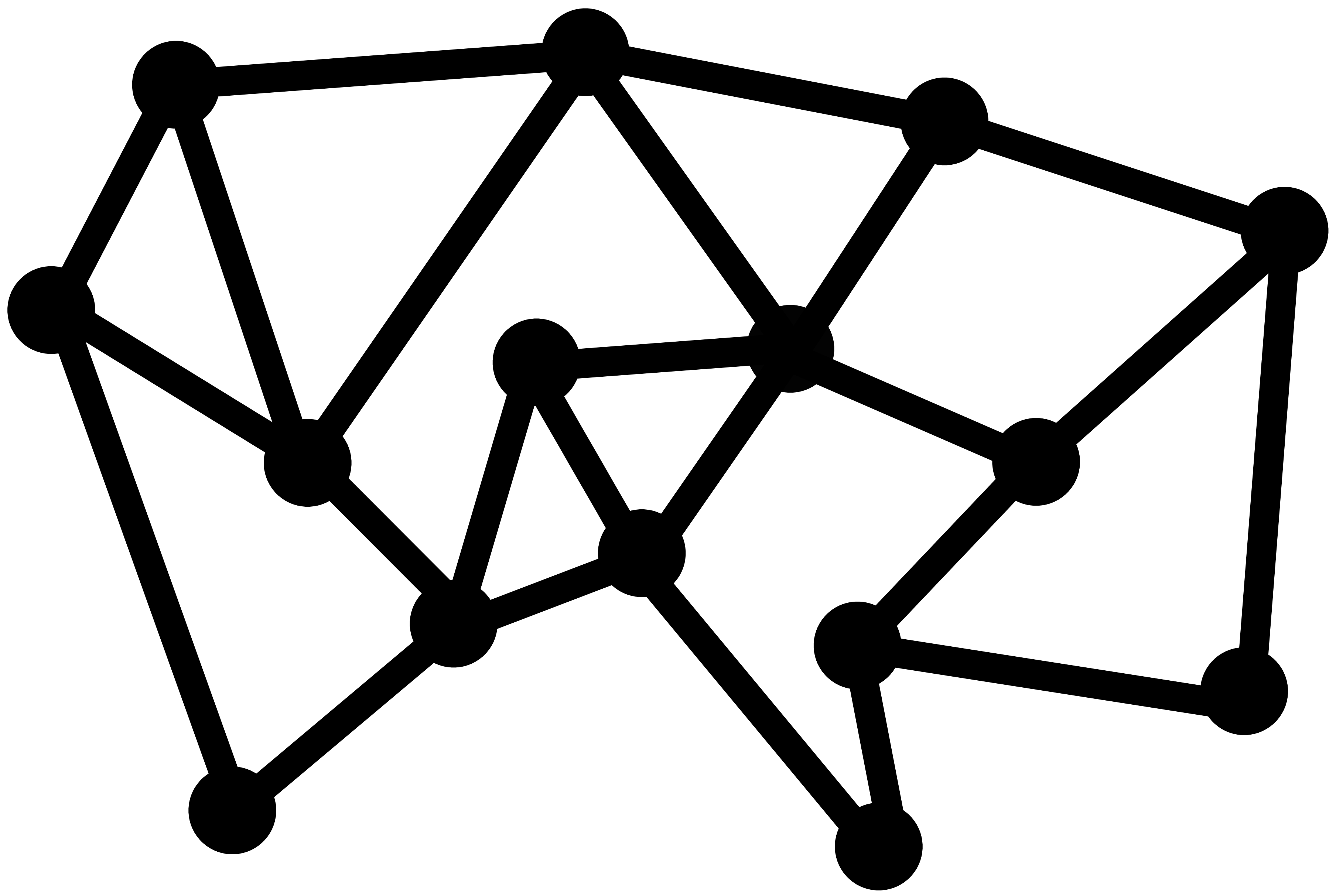
$\forall S \subset V$

2EC NP-Hard

Goal: efficiently find subset of min-weight subgraph satisfying *cut constraints*

Papers Overview

Background: Linear Relaxations

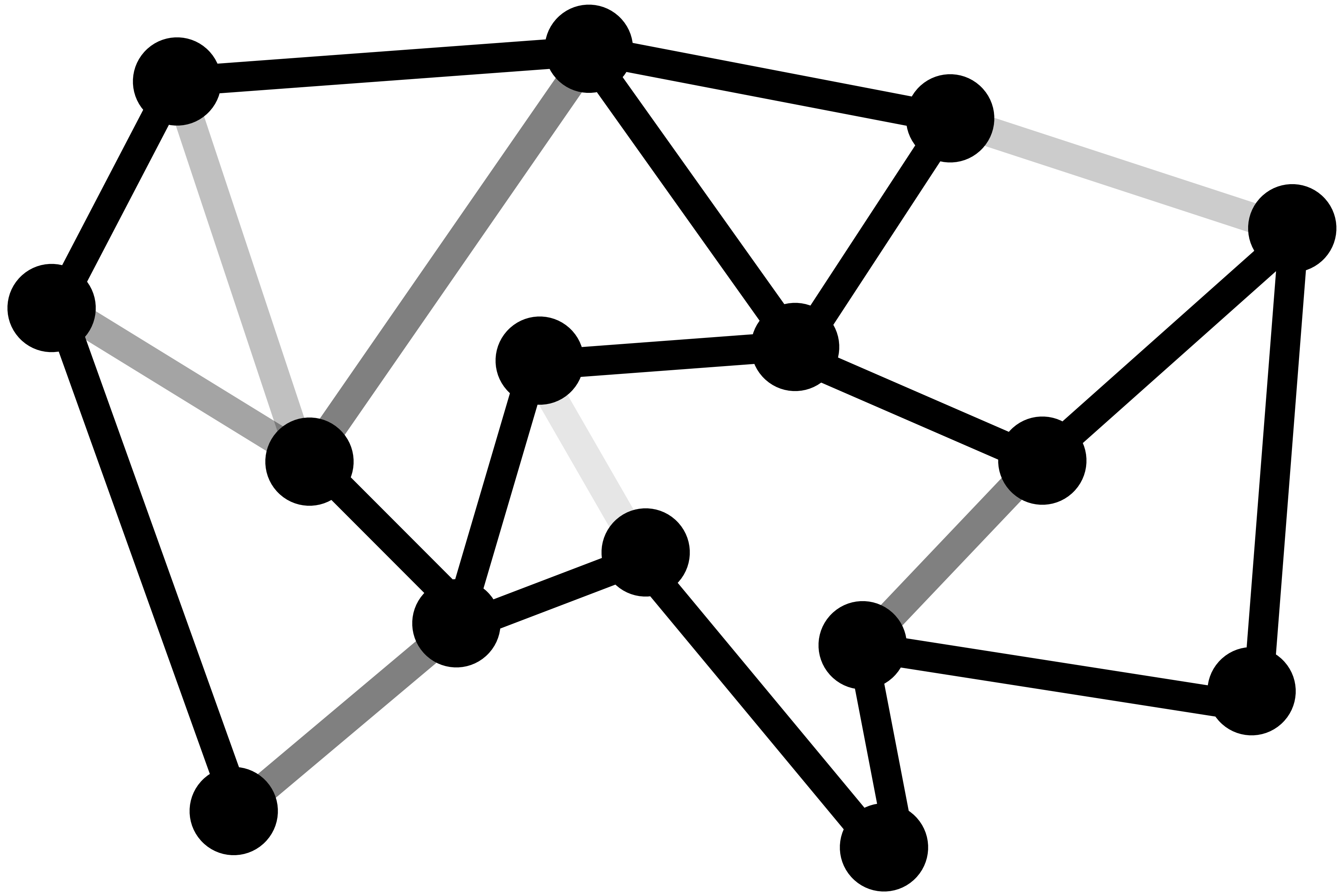


solve problem "fractionally"

Goal: efficiently find subset of min-weight subgraph satisfying *cut constraints*

Papers Overview

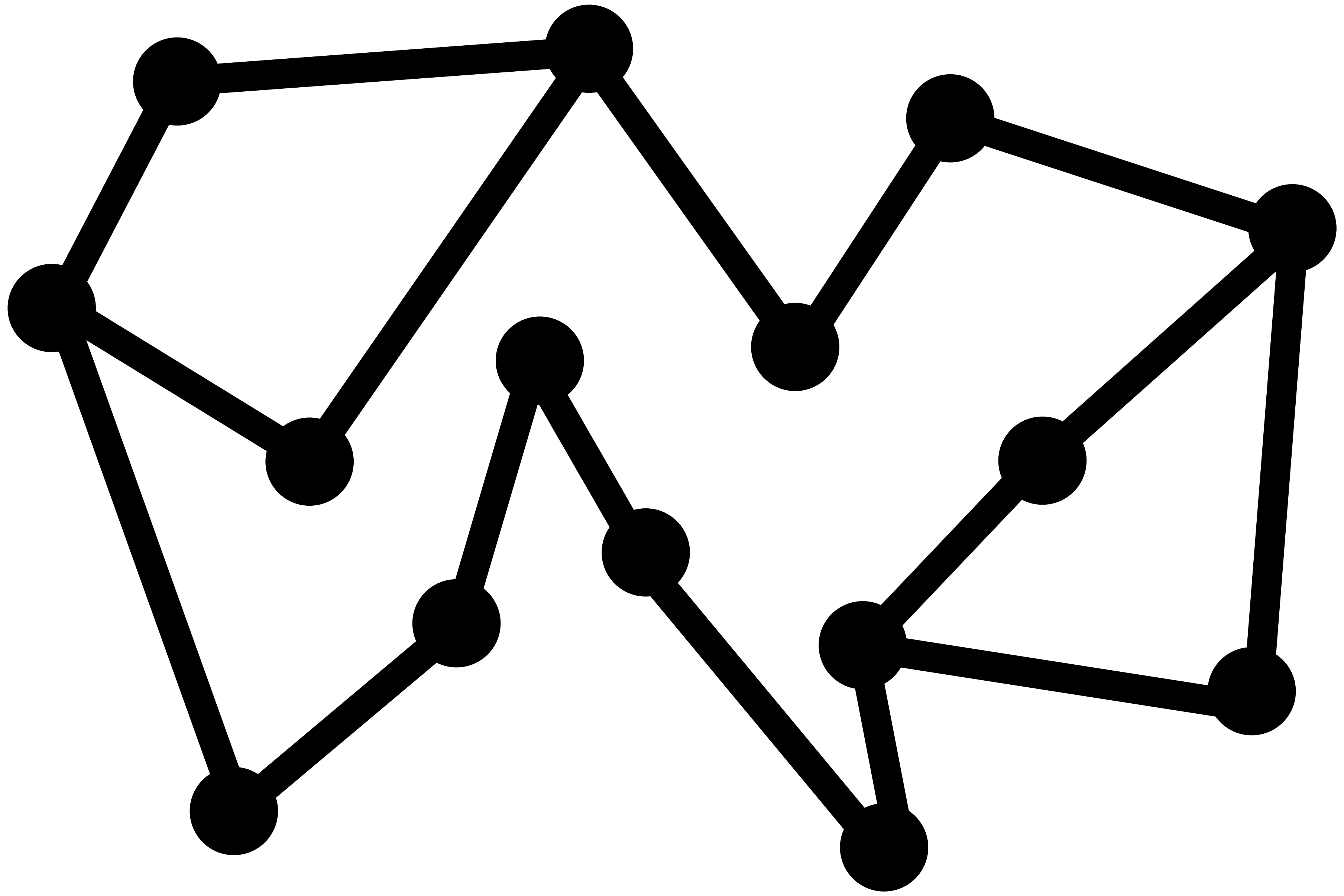
Background: Linear Relaxations



solve problem "fractionally"

$\in P$

use structure

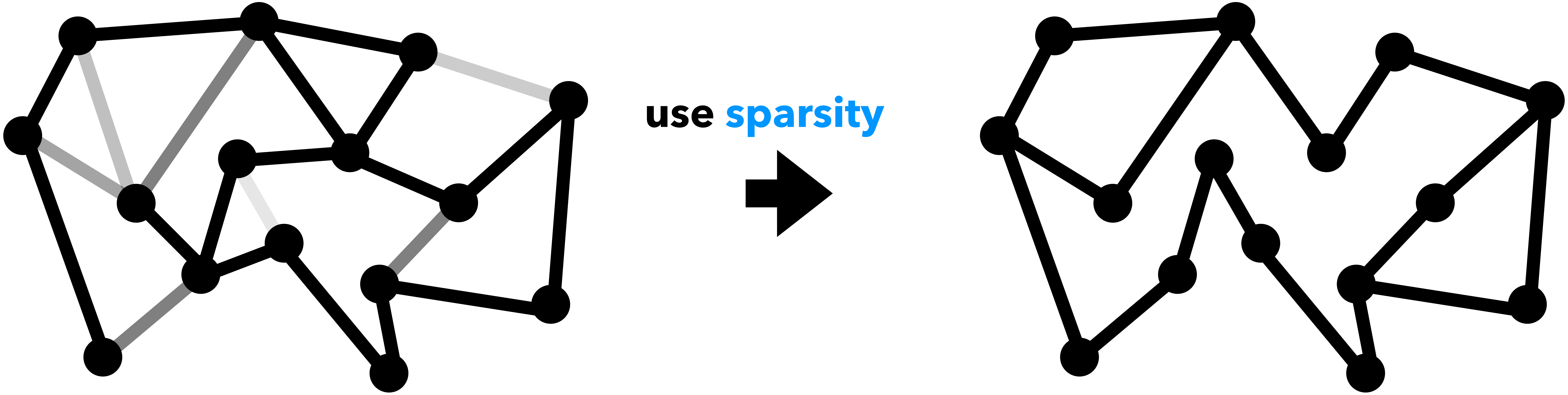


"integral" solution

Goal: efficiently find subset of min-weight subgraph satisfying *cut constraints*

Papers Overview

Background: Linear Relaxations



solve problem "fractionally" $\in P$

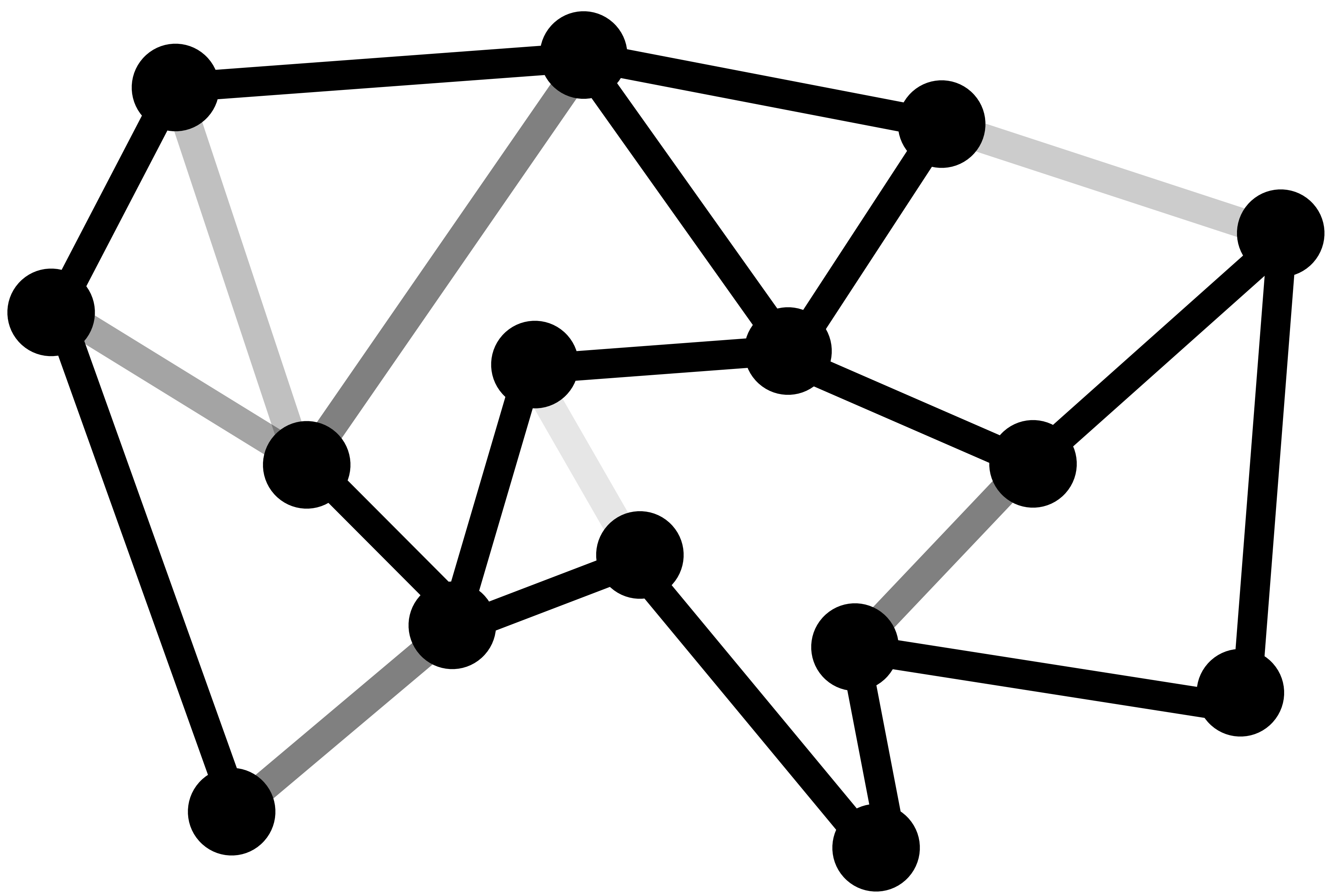
"integral" solution

Goal: efficiently find subset of min-weight subgraph satisfying *cut constraints*

Papers Overview

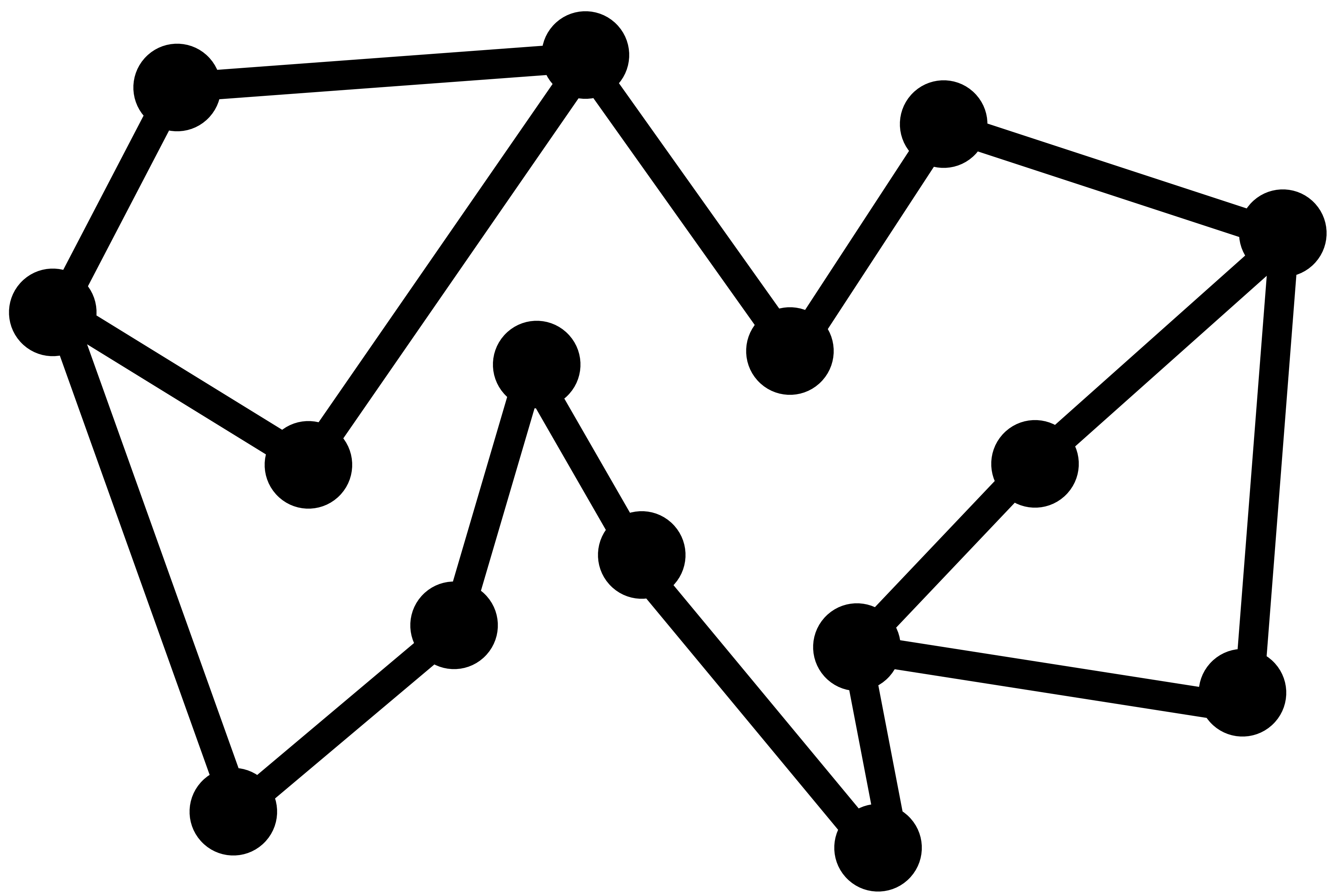
Paper 11: Survivable Network Design

Cool application of LA to combinatorial problem!



solve problem "fractionally"

use sparsity



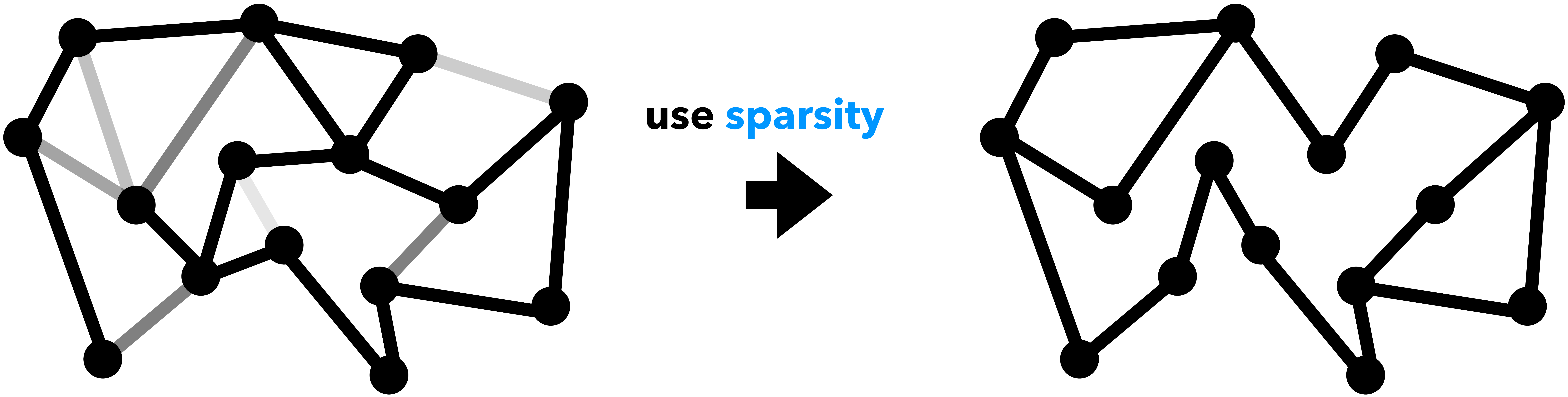
"integral" solution

Theorem 1: for a *general class of ND problems*, can always compute optimal fractional solution with support size $O(n)$

Papers Overview

Paper 11: Survivable Network Design

Cool application of LA to combinatorial problem!



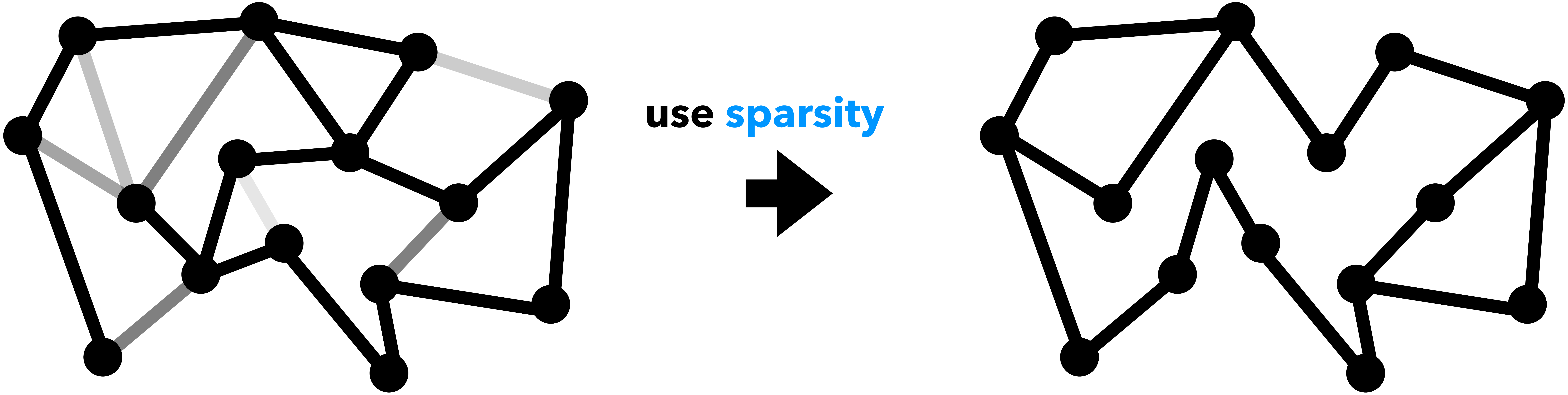
solve problem "fractionally"

"integral" solution

Theorem 2: poly-time 2-approximation for a *general class of ND problems*

Papers Overview

Paper 12: Bounded Degree Spanning Trees



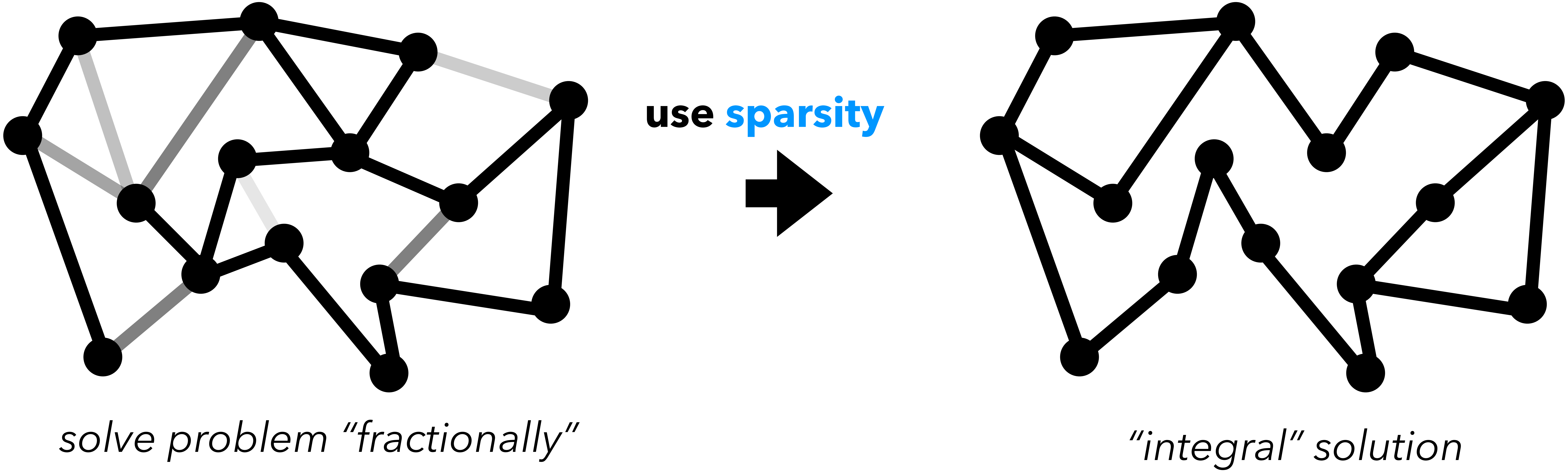
solve problem "fractionally" *"integral" solution*

Theorem 1: for a *general class of ND problems*, can always compute optimal fractional solution with $O(1)$ arboricity (i.e. everywhere sparse)

Most aesthetic paper

Papers Overview

Paper 12: Bounded Degree Spanning Trees

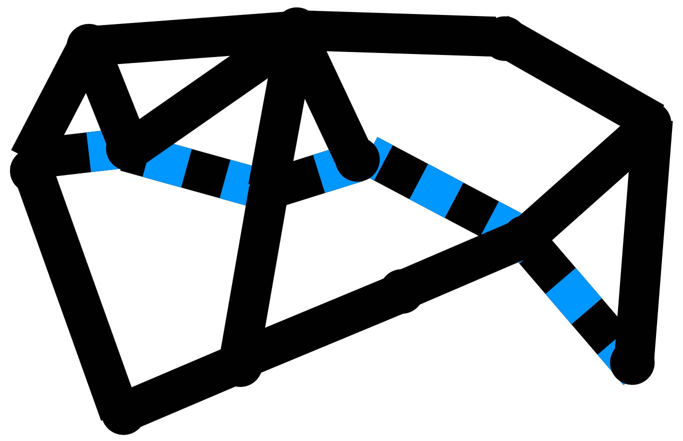


Theorem 2: +2 approximation to degree-bounded spanning tree problem

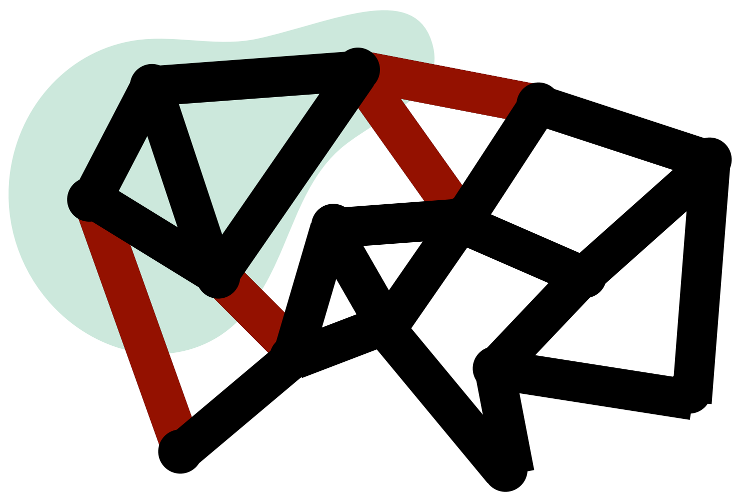
Papers Overview

Sparsification of Five Graph-Theoretic Objects

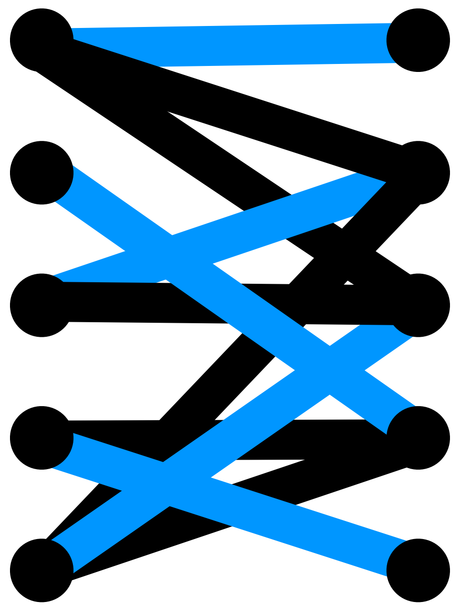
Distances



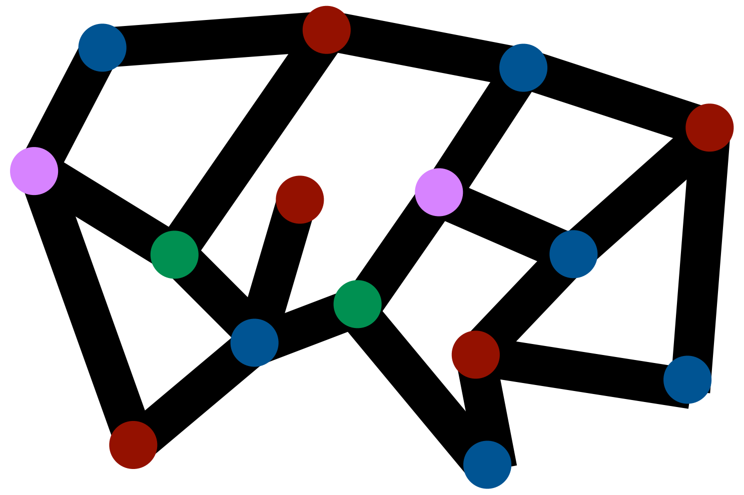
Cuts/Flows



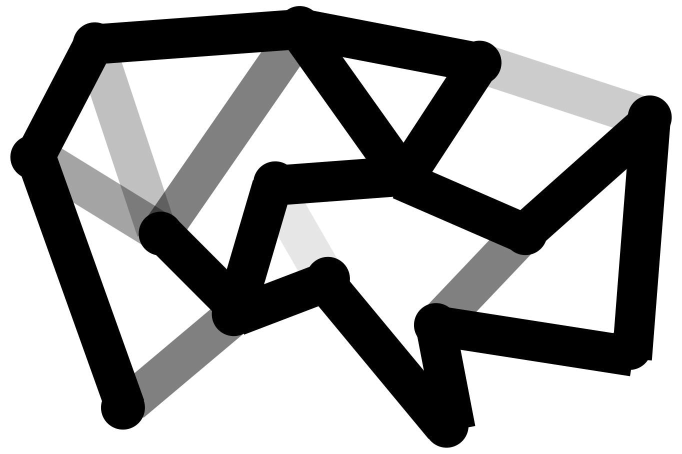
Matchings



Colorings



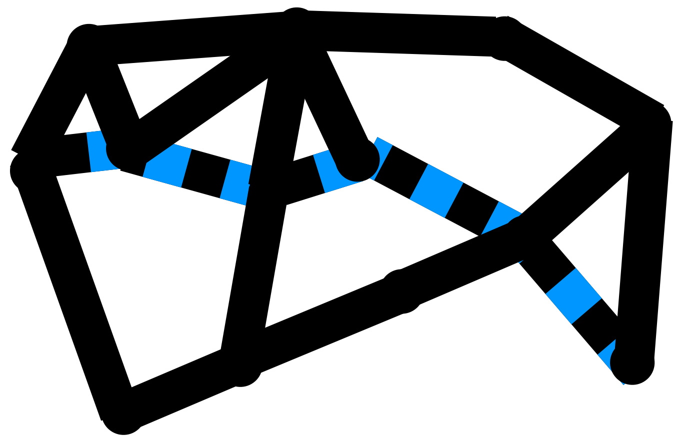
Fractional Opts



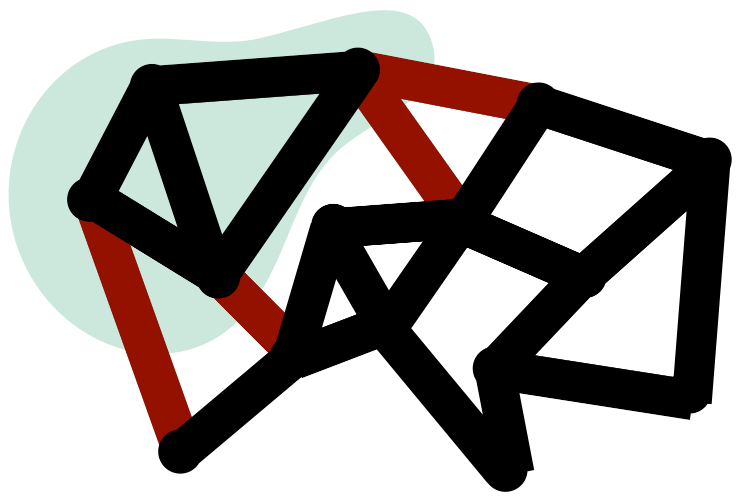
Papers Overview

Sparsification of Five Graph-Theoretic Objects

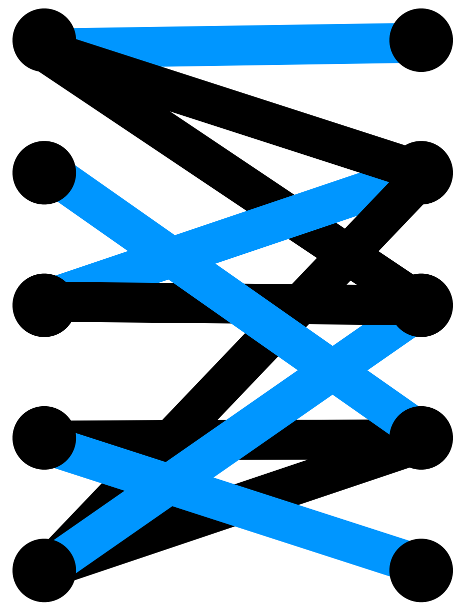
Distances



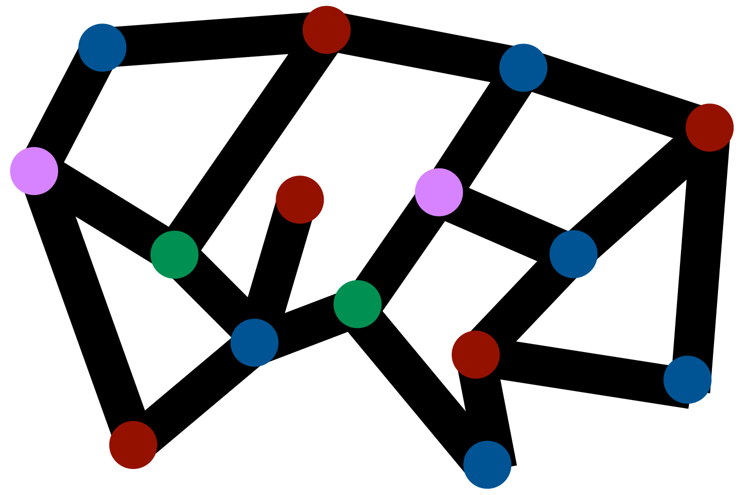
Cuts/Flows



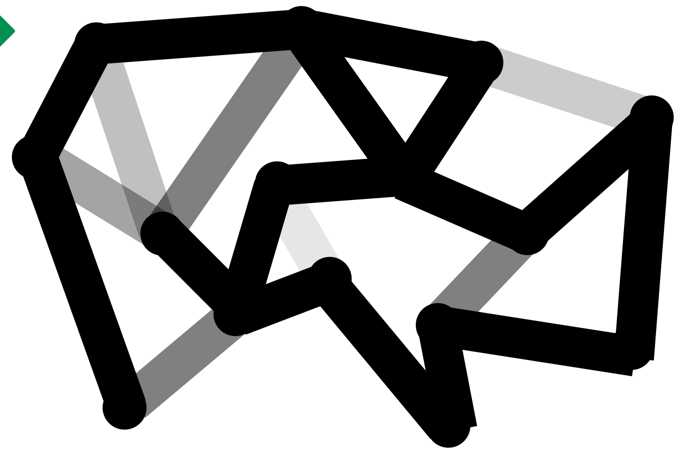
Matchings



Colorings



Fractional Opts



Summary

- **Coming up:**
  - Next week is me (how to read, present, listen to theory and spanners)
  - Following (Sep. 20) is first student talk
  - Will send form with paper preferences for remaining papers after shopping
- **Responsibilities:**
  1. Fill out form of top 3 papers (**need Sep 20, 27 ASAP**)
  2. Read your assigned paper
  3. Prepare talk on paper + 6 questions
  4. Practice (first half of) talk with me week before
  5. Actively participate and give feedback at end of talk