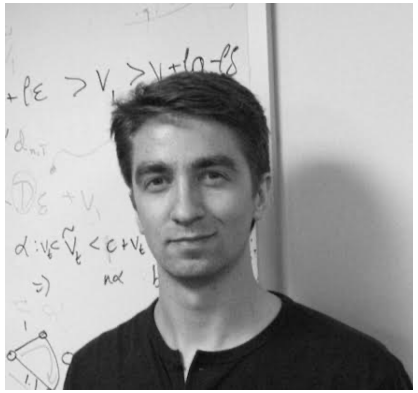
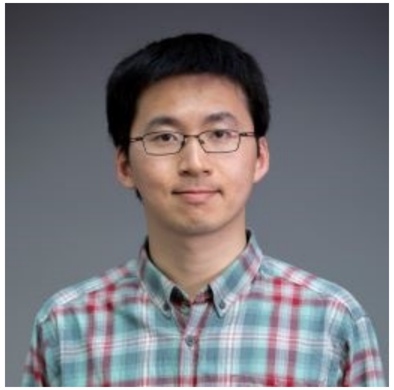
Max Flow and Min-Cost Flow in Almost-Linear Time

Yang P. Liu (Stanford) and Li Chen (Georgia Tech)

Joint with



Rasmus Kyng
ETH



Richard Peng
U. Waterloo

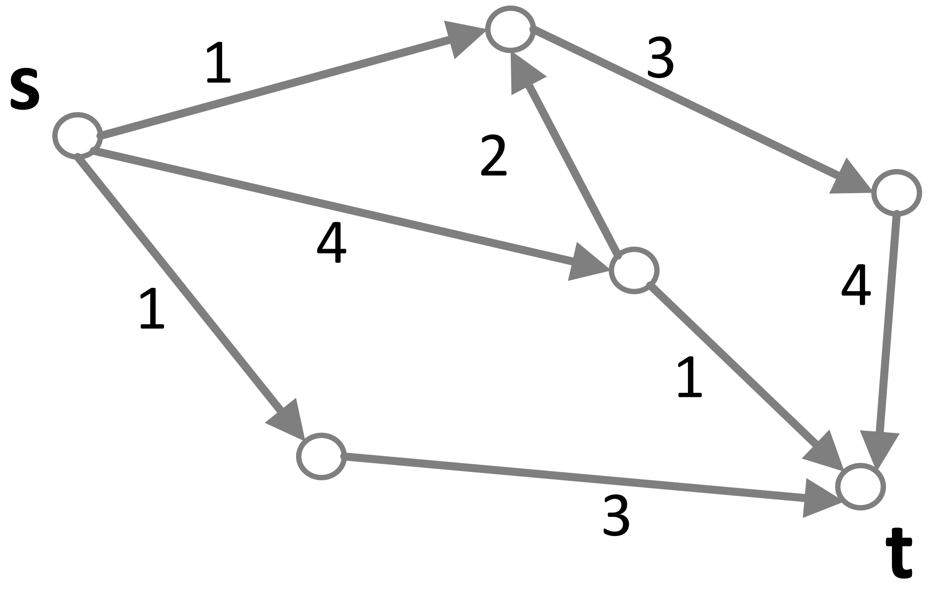
Maximilian
Probst Gutenberg
ETH

Sushant
Sachdeva
U. Toronto

Maximum Flow

Directed graph G = (V, E). m edges, n vertices, source s, sink t

edge *capacities* $u_e \geq 0$, integer in $[0, U]$, where $U = m^{O(1)}$

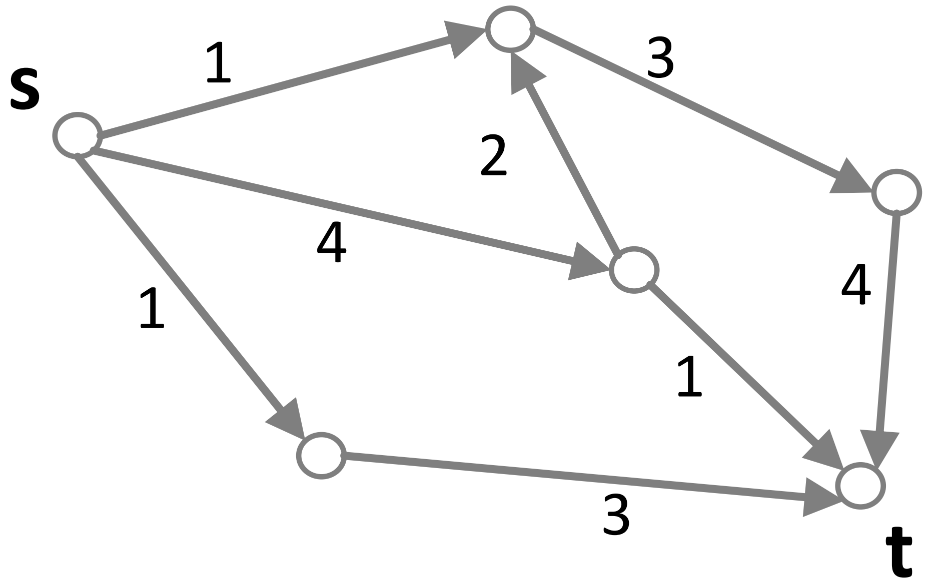


Maximum Flow

Directed graph G = (V, E). m edges, n vertices, source s, sink t

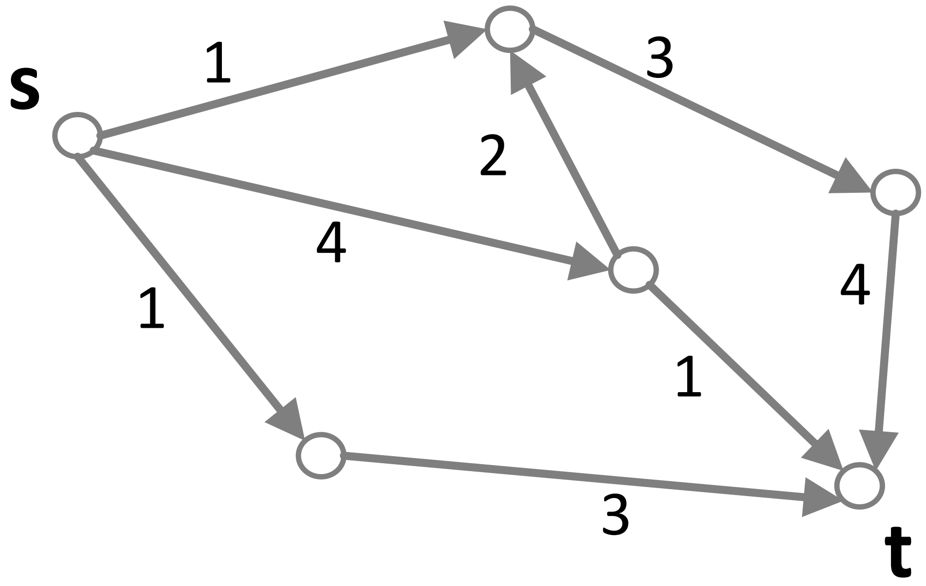
edge *capacities* $u_e \geq 0$, integer in $[0, U]$, where $U = m^{O(1)}$

Goal: Route maximum flow from $s \to t$, Subject to capacities u_e



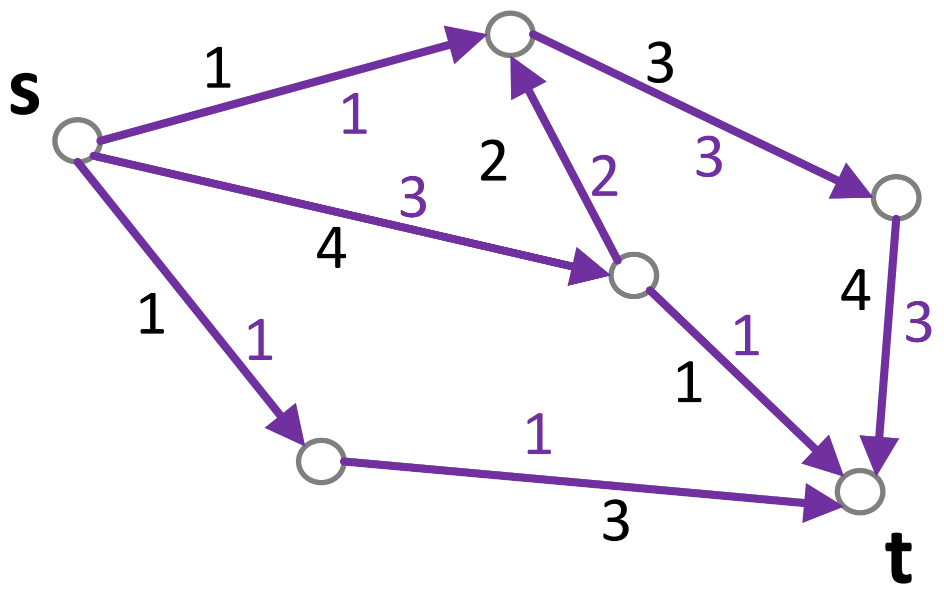
Linear Algebraic View for Max-Flow

$f \in \mathbb{R}^E$, i.e. a real vector on the edges



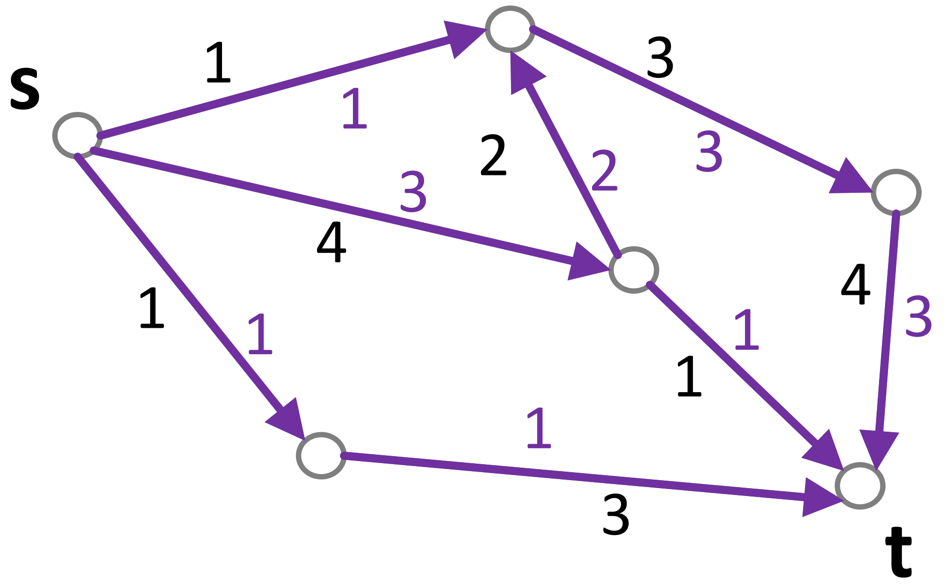
Linear Algebraic View for Max-Flow

$f \in \mathbb{R}^E$, i.e. a real vector on the edges



Linear Algebraic View for Max-Flow

$f \in \mathbb{R}^E$, i.e. a real vector on the edges

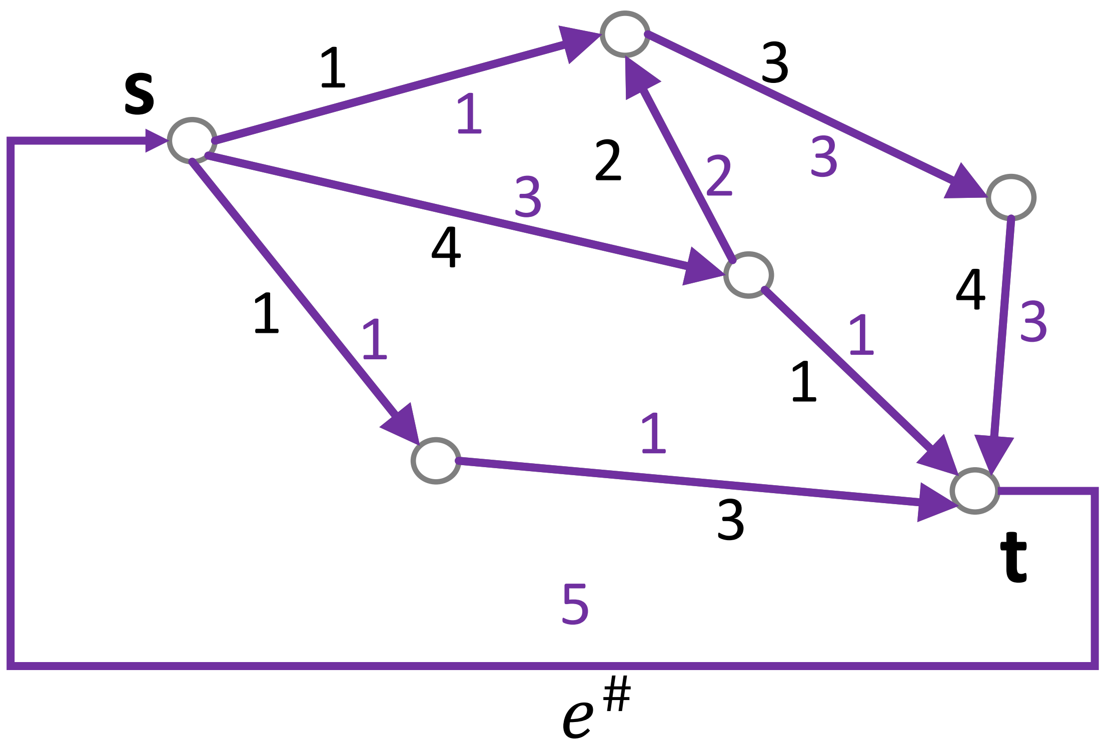


Capacity constraint:

$0 \leq f_e \leq \mathrm{u_e}$

Linear Algebraic View for Max-Flow

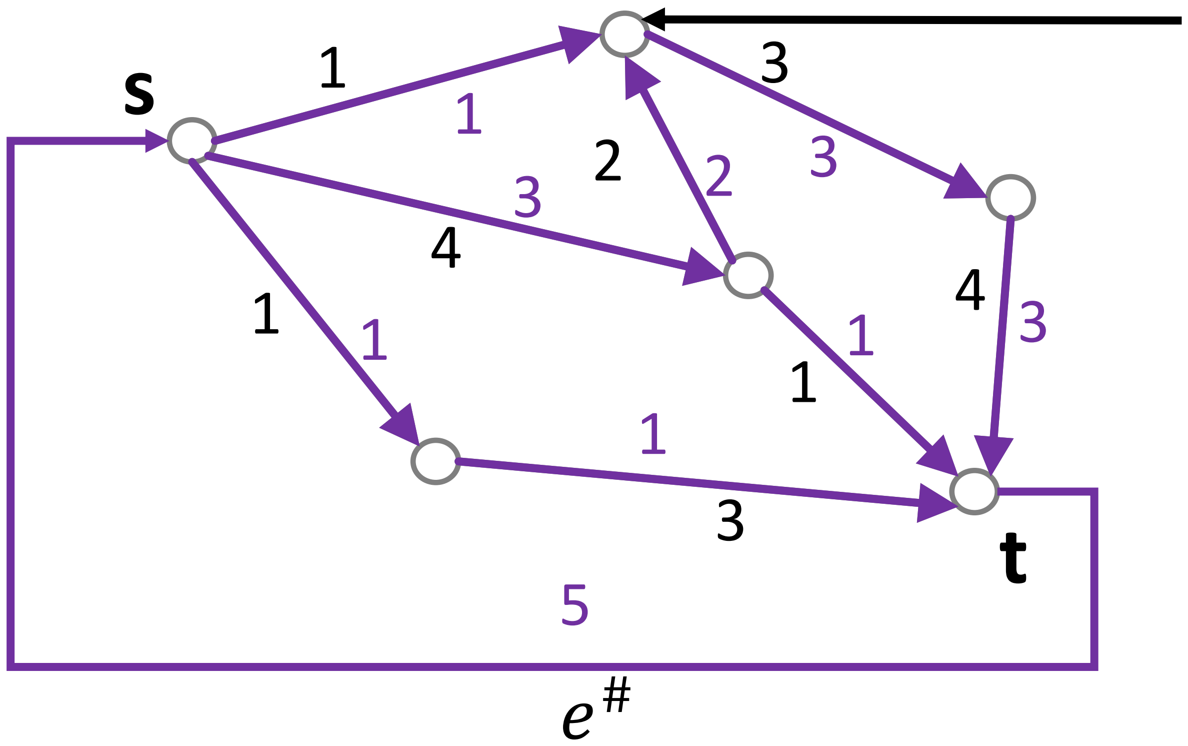
$f \in \mathbb{R}^E$, i.e. a real vector on the edges



Capacity constraint:
$0 \le f_e \le \mathrm{u_e}$

Linear Algebraic View for Max-Flow

$f \in \mathbb{R}^{E}$, i.e. a real vector on the edges



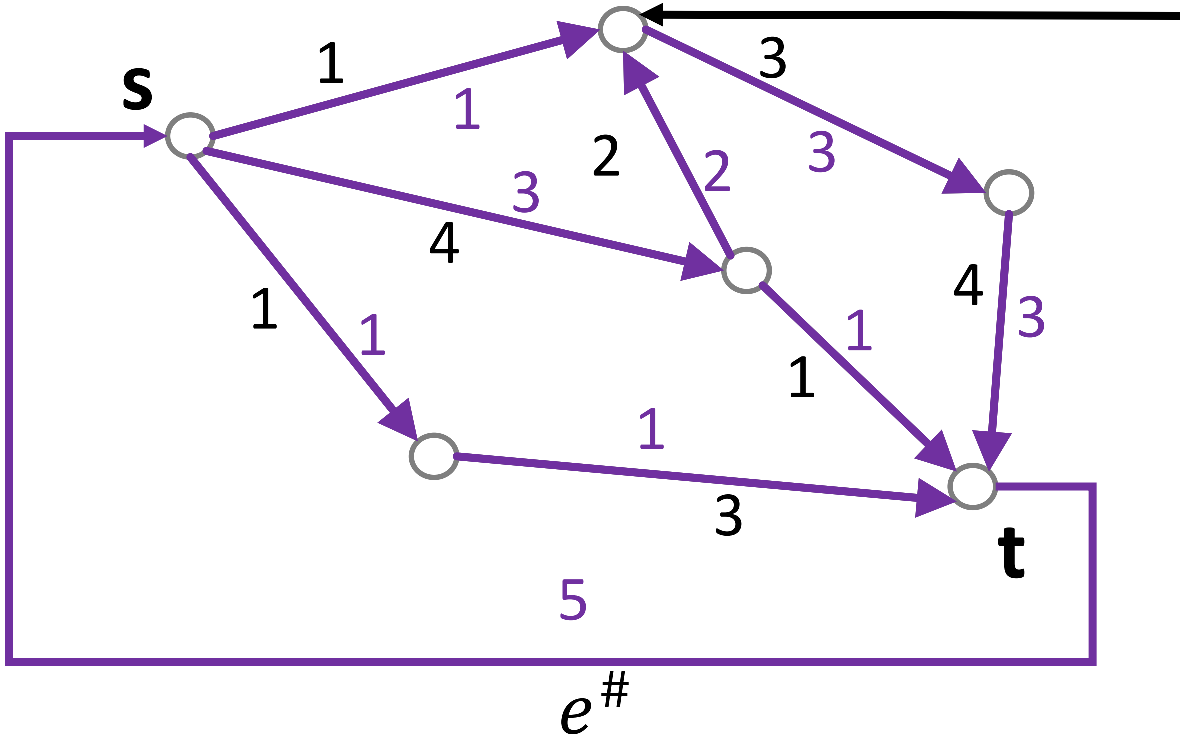
Net zero flow constraint: all vertices have incoming flow=outgoing flow

Capacity constraint: $0 \leq f_e \leq u_e$

Linear Algebraic View for Max-Flow

$f \in \mathbb{R}^E$, i.e. a real vector on the edges

Goal: Route maximum flow on $e^{\#}$



Net zero flow constraint: all vertices have incoming flow=outgoing flow

Capacity constraint: $0 \leq f_e \leq u_e$

Linear Program for Max-Flow

$$\min_{f} \ -f_{e^{\#}}$$

Max flow

For all edges e

$$0 \leq f_e \leq u_e$$

Direction and Capacity constraints

For all vertices x

$$B^T f = 0$$

Net flow constraints

Linear Program for Max-Flow

$$\min_{f} \; -f_{e^{\#}}$$ Max flow

For all edges e $\quad 0 \le f_e \le u_e \quad$ Direction and Capacity constraints

For all vertices x $\quad B^T f = 0 \quad$ Net flow constraints

[C-Kyng-L-Peng-Probst Gutenberg-Sachdeva]

Can solve max-flow in $m^{1+o(1)}$ time

General Convex Flow Program

$$\min_f \sum_e cost_e(f_e)$$ Flow Cost

For all edges e $$0 \leq f_e \leq u_e$$ Direction and Capacity constraints

For all vertices x $$B^T f = d$$ Net flow constraints

[C-Kyng-L-Peng-Probst Gutenberg-Sachdeva]

Can solve general convex* flows in $m^{1+o(1)}$ time

*(assuming costs are specified as efficient self-concordant functions)

Applications: Almost-Linear time Algorithms

(Min-cost) Bi-partite matching

Min-cost flow

Negative weight shortest paths

Worker assignment

Optimal Transport

Directed flows with vertex capacities / costs

Undirected vertex connectivity

Flow diffusion

…

Applications: Almost-Linear time Algorithms

(Min-cost) Bi-partite matching
Min-cost flow
Negative weight shortest paths
Worker assignment
Optimal Transport
Directed flows with vertex capacities / costs
Undirected vertex connectivity
Flow diffusion
…

Matrix Scaling
Isotonic Regression
Weighted p-norm Flows
Entropic-regularized Optimal Transport
…

Comparison to Previous Works

[Ford-Fulkerson`54]
$\tilde{O}(m^2 U)$

[Galil-Naamad`80, Sleator-Tarjan`83]
$\tilde{O}(mn \log n)$

[Goldberg-Rao`98]
$\tilde{O}(m^{3/2})$

Combinatorial outer and inner algorithms

Comparison to Previous Works

[Ford-Fulkerson`54] $\tilde{O}(m^2 U)$

[Galil-Naamad`80, Sleator-Tarjan`83] $\tilde{O}(mn \log n)$

[Goldberg-Rao`98] $\tilde{O}(m^{3/2})$

[Daitch-Spielman`08] $\tilde{O}(m^{3/2})$

[Madry`13] $\tilde{O}(m^{10/7} U^{1/7})$

[Lee-Sidford`14] $\tilde{O}(mn^{1/2})$

[Kathuria-L-Sidford`20] $m^{4/3+o(1)} U^{1/3}$

[Brand et al `20, Brand et al `21] $\tilde{O}(m + n^{3/2})$

[Gao-L-Peng`21, Brand et al `21] $\tilde{O}(m^{3/2-1/58})$

Combinatorial outer and inner algorithms

$O(\sqrt{m})$ or better iteration IPM based outer algorithm + $\Omega(m)$-time numerical inner algorithm (ℓ_2, ℓ_p flows)

Improved IPM

Improved inner loop via dynamic algorithms

Comparison to Previous Works

[Ford-Fulkerson`54] $\tilde{O}(m^2 U)$

[Galil-Naamad`80, Sleator-Tarjan`83] $\tilde{O}(mn \log n)$

[Goldberg-Rao`98] $\tilde{O}(m^{3/2})$

[Daitch-Spielman`08] $\tilde{O}(m^{3/2})$

[Madry`13] $\tilde{O}(m^{10/7} U^{1/7})$

[Lee-Sidford`14] $\tilde{O}(mn^{1/2})$

[Kathuria-L-Sidford`20] $m^{4/3+o(1)} U^{1/3}$

[Brand et al `20, Brand et al `21] $\tilde{O}(m + n^{3/2})$

[Gao-L-Peng`21, Brand et al `21] $\tilde{O}(m^{3/2-1/58})$

This work $m^{1+o(1)}$

Combinatorial outer and inner algorithms

$O(\sqrt{m})$ or better iteration IPM based outer algorithm + $\Omega(m)$-time numerical inner algorithm (ℓ_2, ℓ_p flows)

Improved IPM

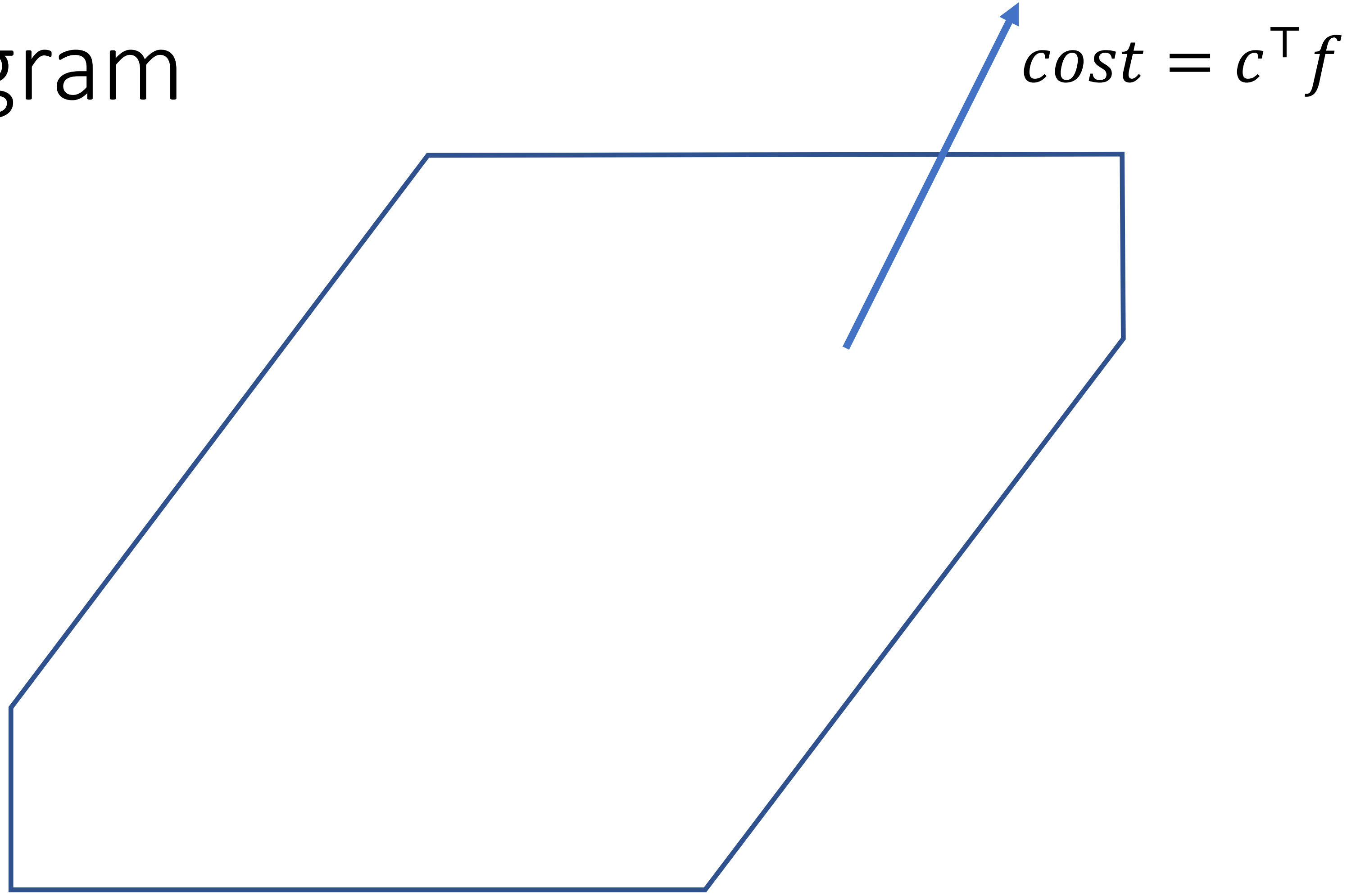
Improved inner loop via dynamic algorithms

$m^{1+o(1)}$ iteration L1 IPM
+
combinatorial inner problem
+
$m^{o(1)}$ amortized time dynamic algorithm for inner problem

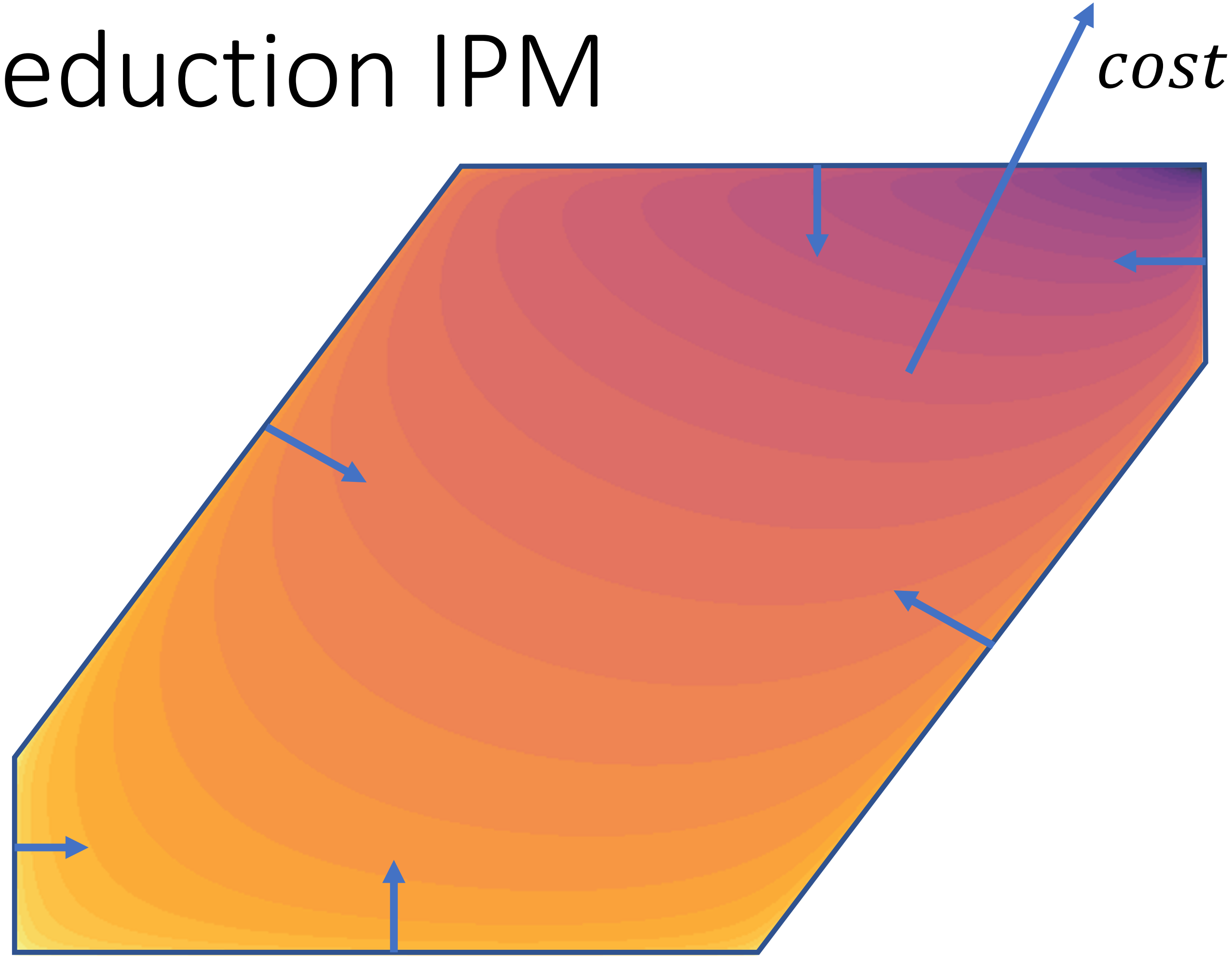
Key Ingredient I:
L1 Interior Point Method (IPM)

Outer Algorithm

Linear Program



Potential Reduction IPM



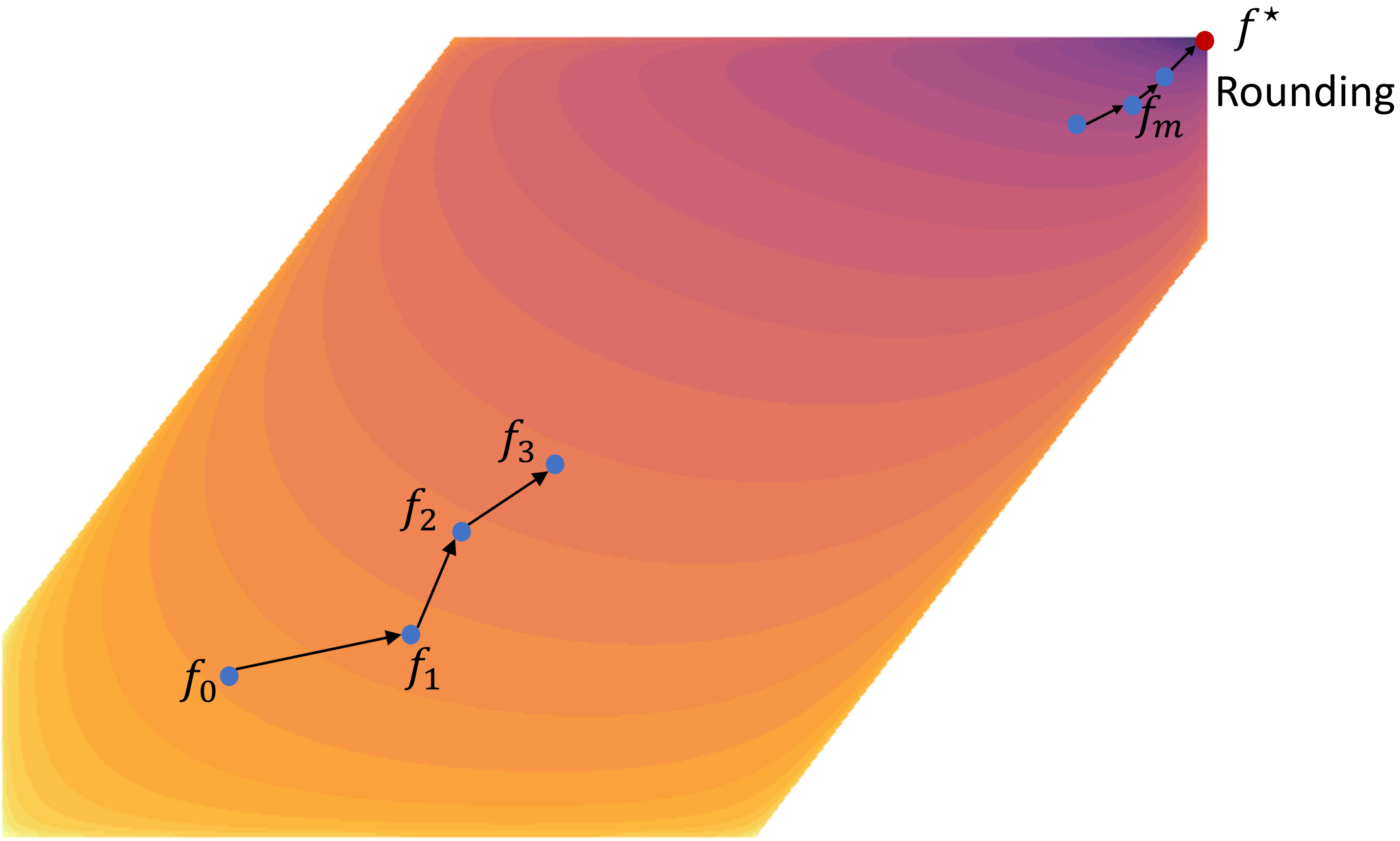
[Karmarkar84]

$$\Phi(f) = m \log(c^\top f - OPT) - \sum_e (\log f_e + \log(u_e - f_e))$$

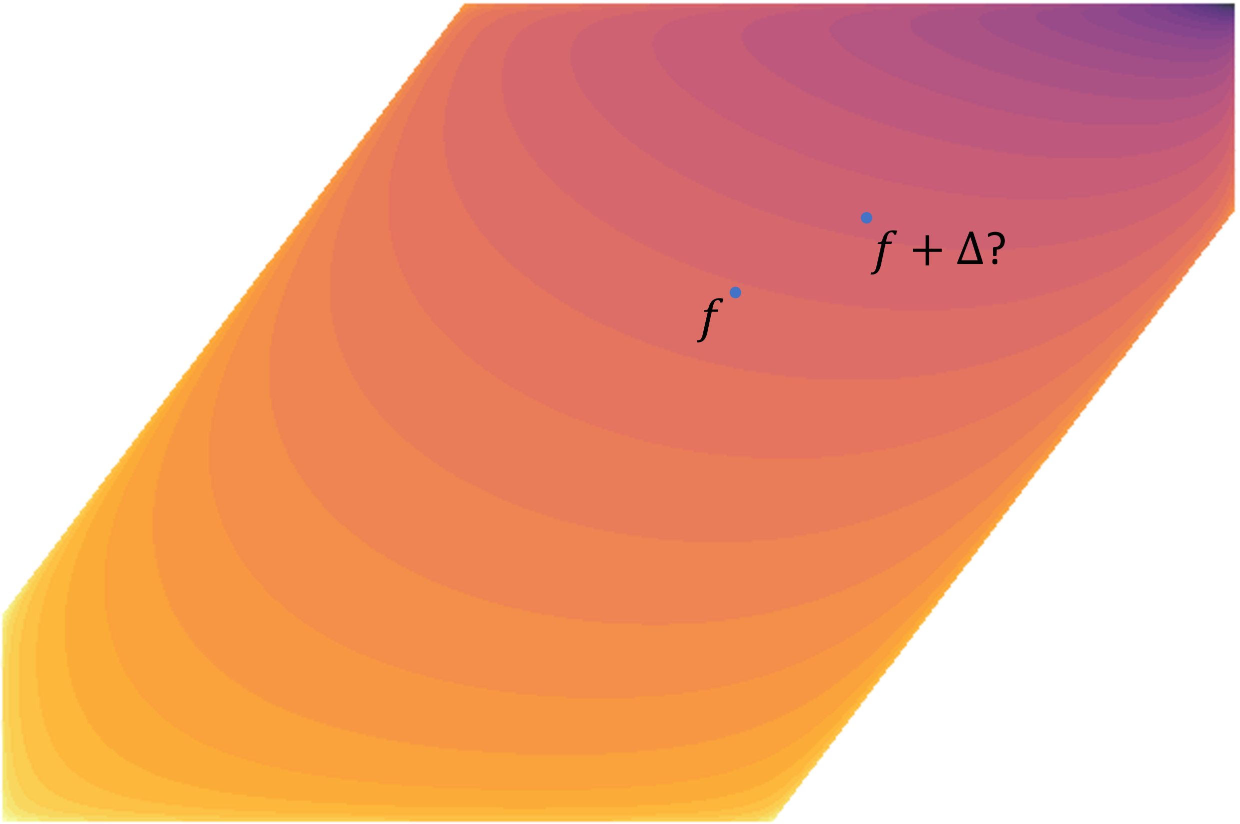
Improves objective

Keep it "far" from hard constraints

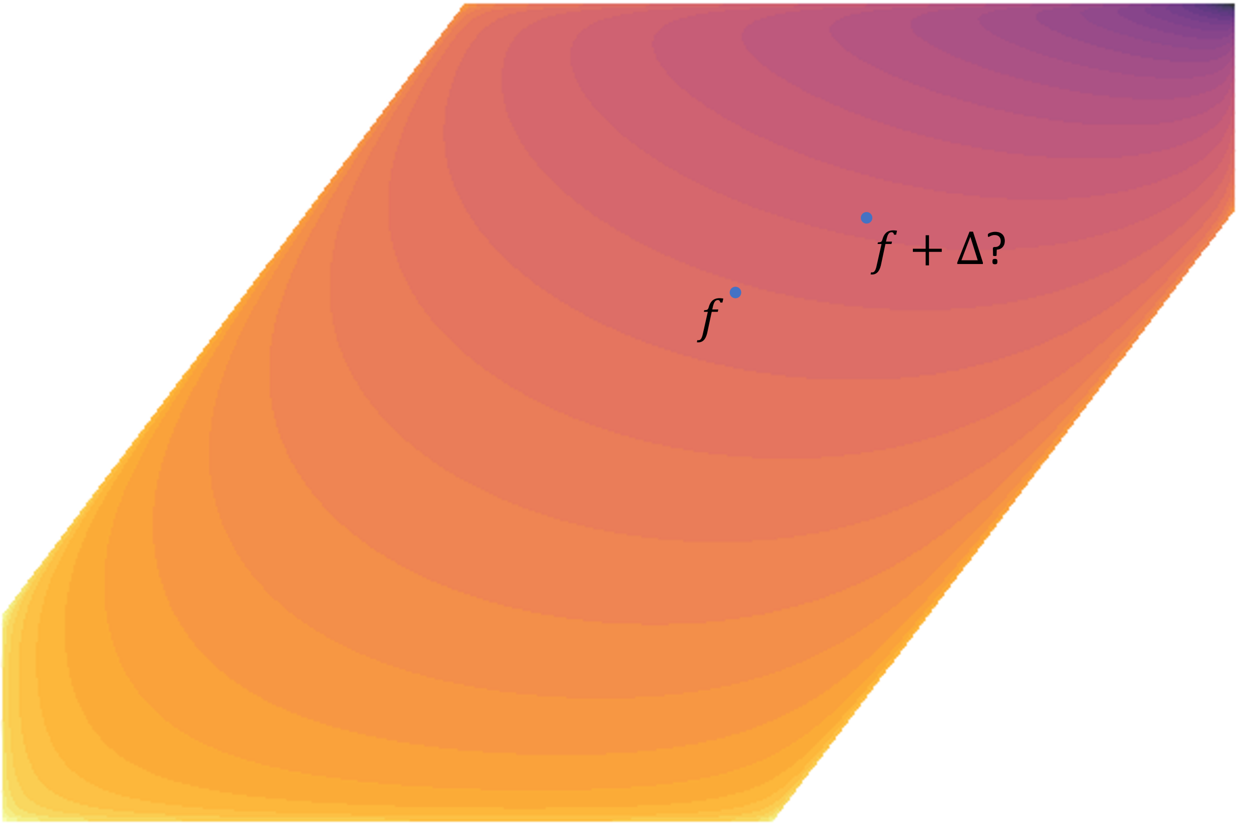
Potential Reduction IPM



Potential Reduction IPM



Potential Reduction IPM



$$\Phi(f + \Delta) \leq \Phi(f) + g^\top \Delta + \|L\Delta\|_2^2$$

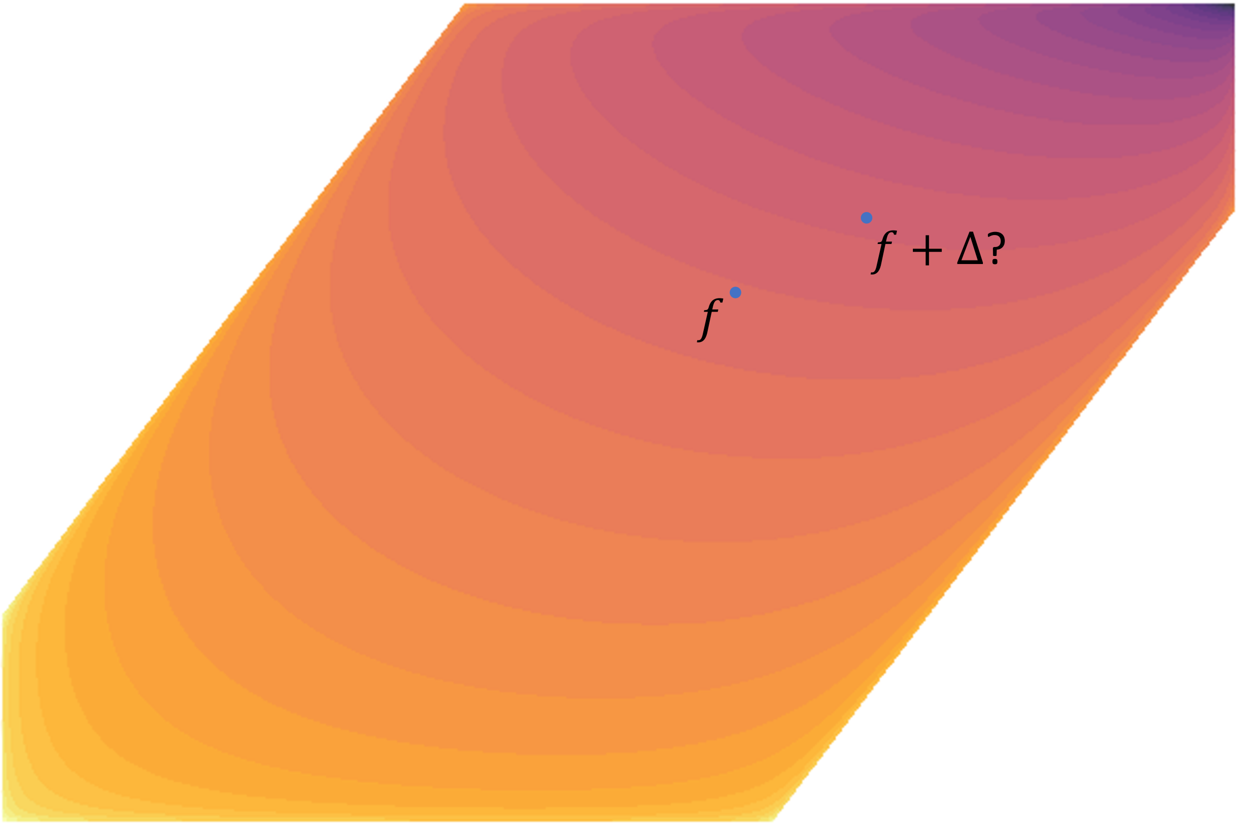
2nd order Taylor expansion

$$g = \nabla\Phi$$

$$L_e = \frac{1}{\min(u_e - f_e, f_e)}$$

Symmetrized residual capacity

Potential Reduction IPM



Electrical flows!

$$\min_{B^\top \Delta = 0} \frac{g^\top \Delta}{\|L\Delta\|_2}$$

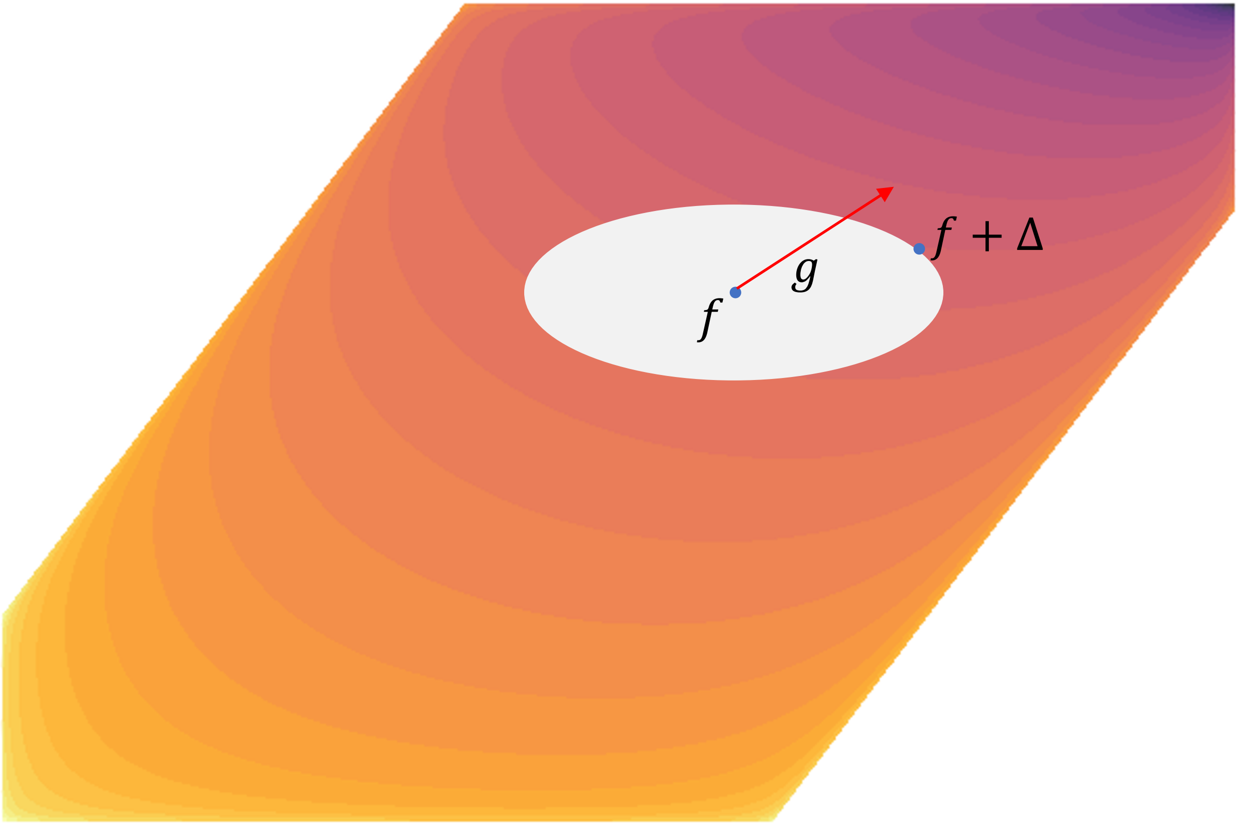
$$g = \nabla \Phi$$

$$L_e = \frac{1}{\min(u_e - f_e, f_e)}$$

$$\Phi(f + \Delta) \leq \Phi(f) + g^\top \Delta + \|L\Delta\|_2^2$$

minimize over circulations $\Delta : B^\top \Delta = 0$

Potential Reduction IPM



Electrical flows!

$$\min_{B^\top \Delta = 0} \frac{g^\top \Delta}{\|L\Delta\|_2}$$

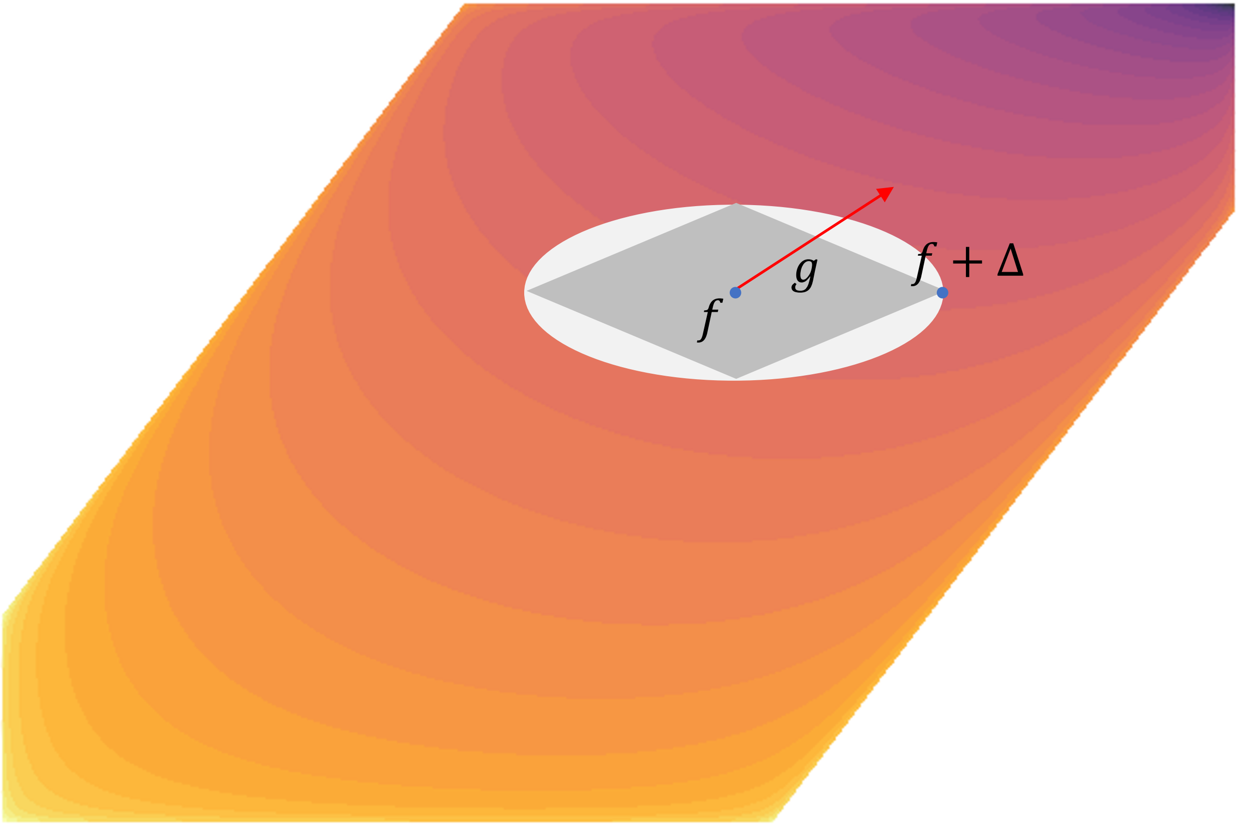
$$\Phi(f + \Delta) \leq \Phi(f) + g^\top \Delta + \|L\Delta\|_2^2$$

minimize over circulations $\Delta : B^\top \Delta = 0$

$$g = \nabla \Phi$$

$$L_e = \frac{1}{\min(u_e - f_e, f_e)}$$

L1 IPM



Min Ratio Cycle

$$\min_{B^\top \Delta = 0} \frac{g^\top \Delta}{\|L\Delta\|_1}$$

$$\Phi(f+\Delta) \le \Phi(f) + g^\top \Delta + \|L\Delta\|_1^2$$

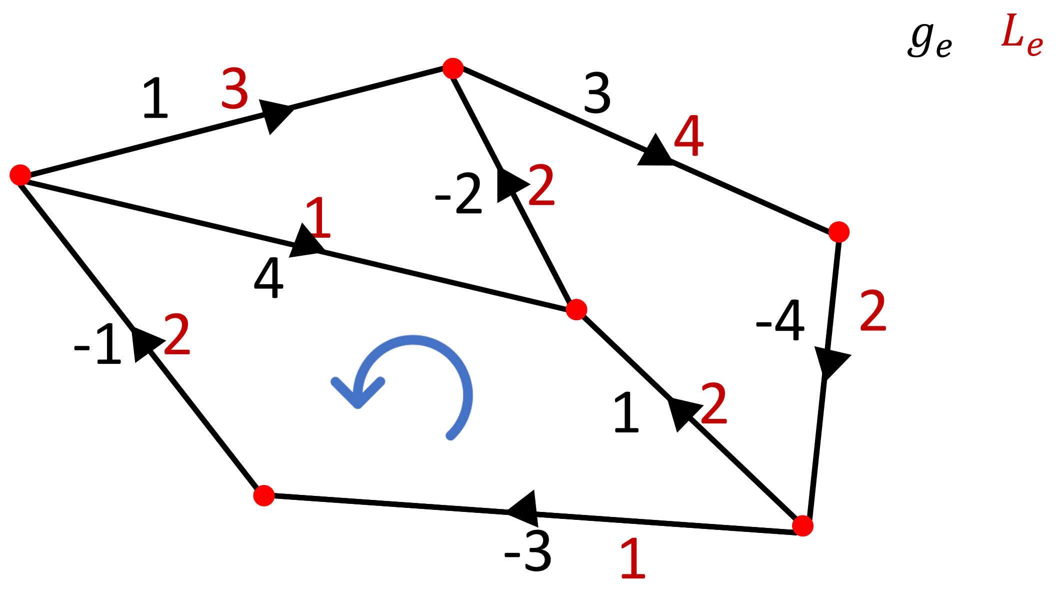
minimize over circulations $\Delta : B^\top \Delta = 0$

$$g = \nabla \Phi$$

$$L_e = \frac{1}{\min(u_e - f_e, f_e)}$$

Min-ratio Cycle

$$\min_{B^\top \Delta = 0} \frac{g^\top \Delta}{\|L\Delta\|_1}$$

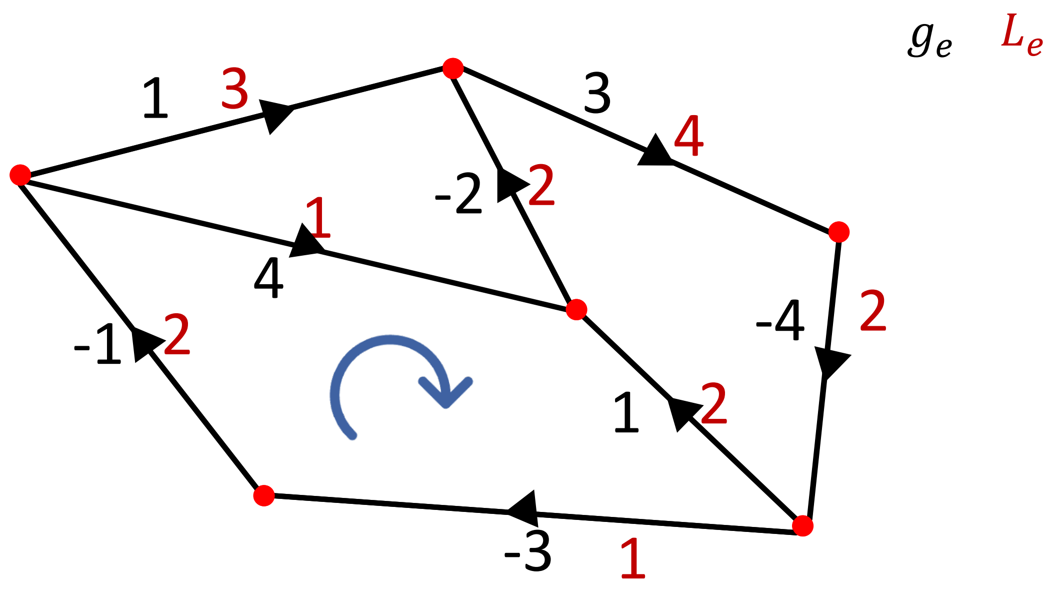


$$\|L\Delta\|_1 = 1 + 2 + 1 + 2 = 6$$

$$g^\top \Delta = -4 + 1 + 3 + 1 = 1$$

Min-ratio Cycle

$$\min_{B^\top \Delta = 0} \frac{g^\top \Delta}{\|L\Delta\|_1}$$



Edges and lengths are undirected

Gradient has a direction

$$\|L\Delta\|_1 = 1 + 2 + 1 + 2 = 6$$

$$g^\top \Delta = 4 - 1 - 3 - 1 = -1$$

Optimal solution can be assumed to be a simple cycle

L1 IPM

[C-Kyng-L-Peng-Probst Gutenberg-Sachdeva]

There is an IPM for max-flow such that

1. $m^{1+o(1)}$ iterations, each subproblem a min-ratio cycle $\min_{B^\top \Delta = 0} \frac{g^\top \Delta}{\|L\Delta\|_1}$
2. a $m^{o(1)}$-approximate solution suffices at each iteration
3. At most $m^{1+o(1)}$ total changes to g_e, L_e over all edges e
4. For each min-ratio cycle problem, $\frac{g^\top (f^\star - f)}{\|L(f^\star - f)\|_1} \leq -0.1$

Key Ingredient II: Min-ratio Cycle Data-Structure

Inner Algorithm

Approx min-ratio cycle via tree embeddings

Goal: Approximately solve $\min_{B^\top \Delta = 0} \frac{g^\top \Delta}{\|L\Delta\|_1}$

Algorithm:

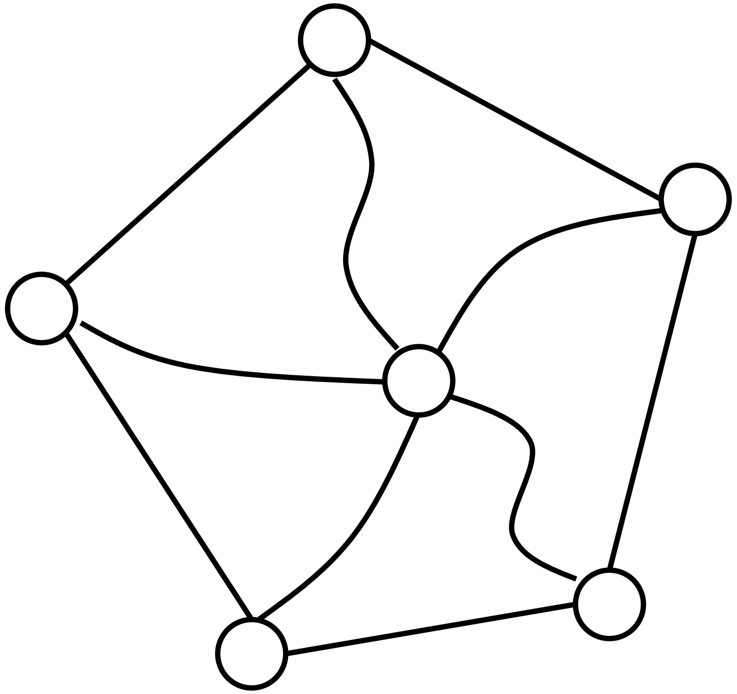
$\tilde{O}(m)$ time

1. Sample a random "low-stretch spanning tree" T
   [Alon-Karp-Peleg-West '95, Elkin-Emek-Spielman-Teng '05, Abraham-Bartal-Neiman '09]
2. Return the best "tree cycle" in T (one off-tree edge + tree path)

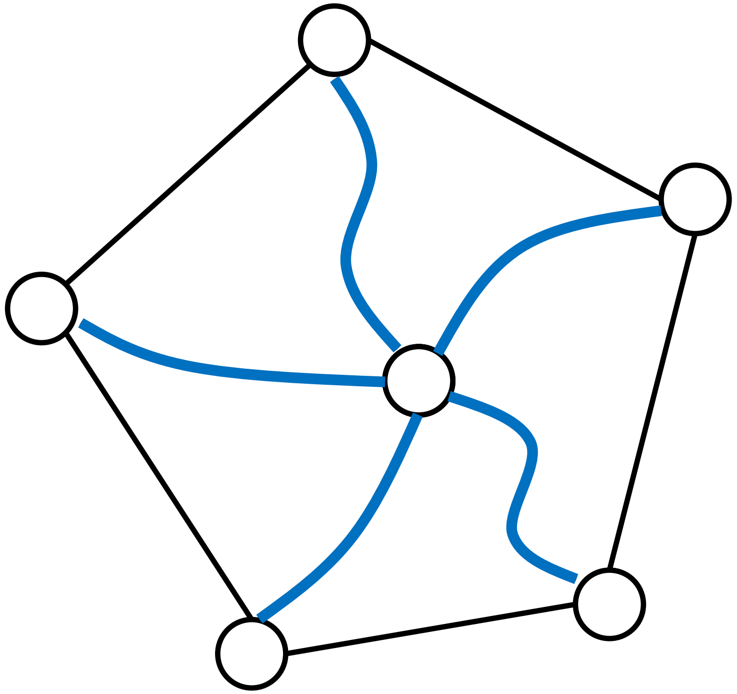
a.k.a. fundamental cycles
Denoted $\mathrm{cycle}_T(e)$

$\tilde{O}(m)$ time

Claim: Some $\mathrm{cycle}_T(e)$ is an $\tilde{O}(1)$-approx

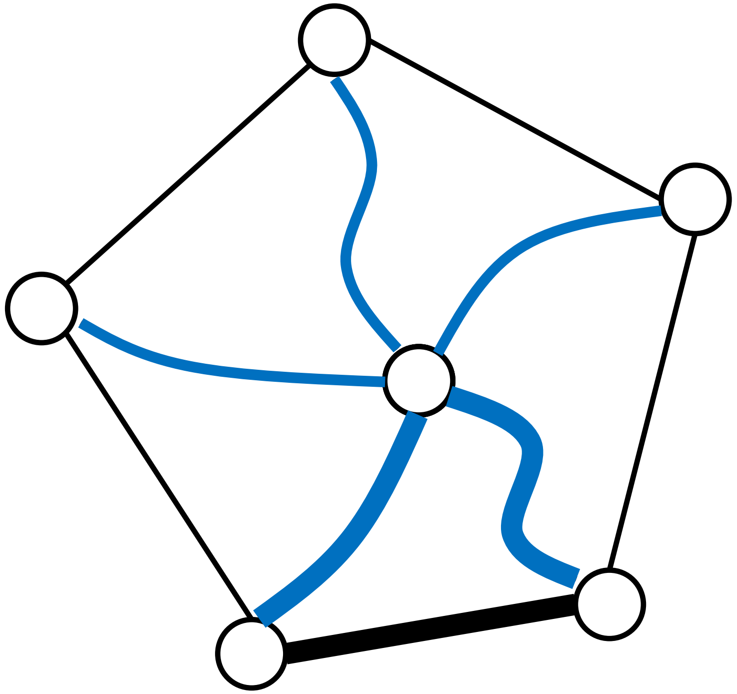


Claim: Some $\mathrm{cycle}_T(e)$ is an $\tilde{O}(1)$-approx



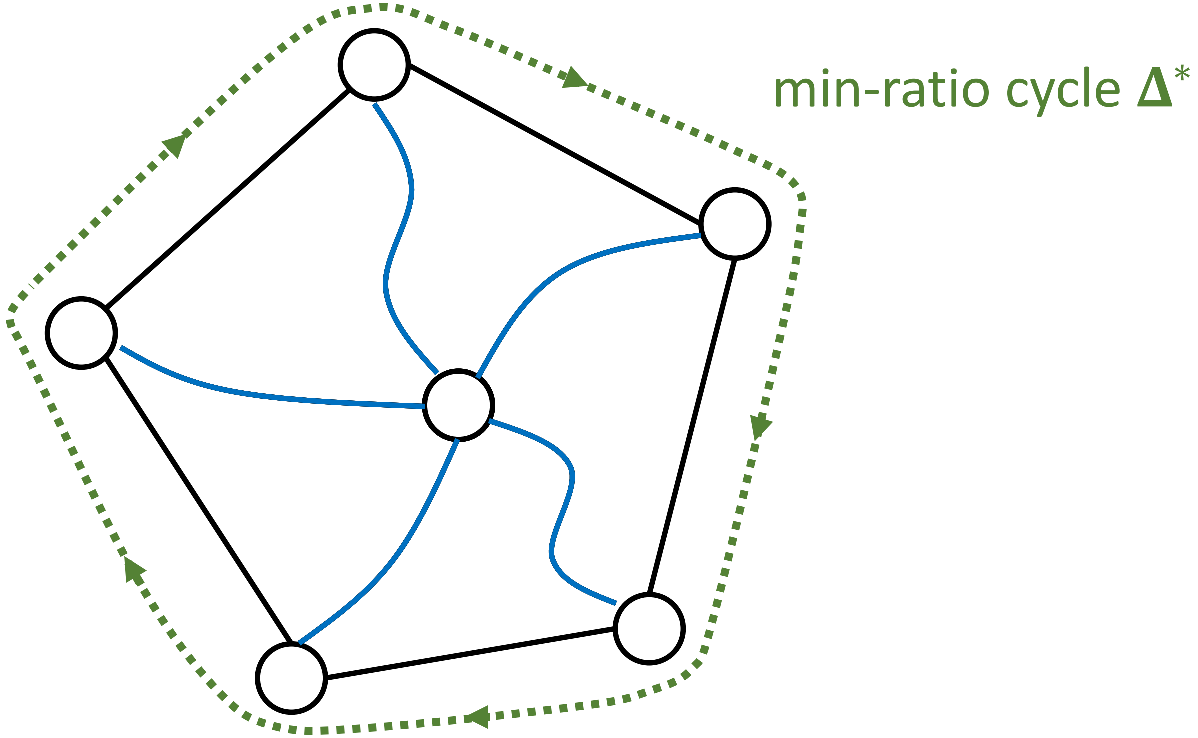
Sample a Low-Stretch tree T

Claim: Some $\mathrm{cycle}_T(e)$ is an $\tilde{O}(1)$-approx



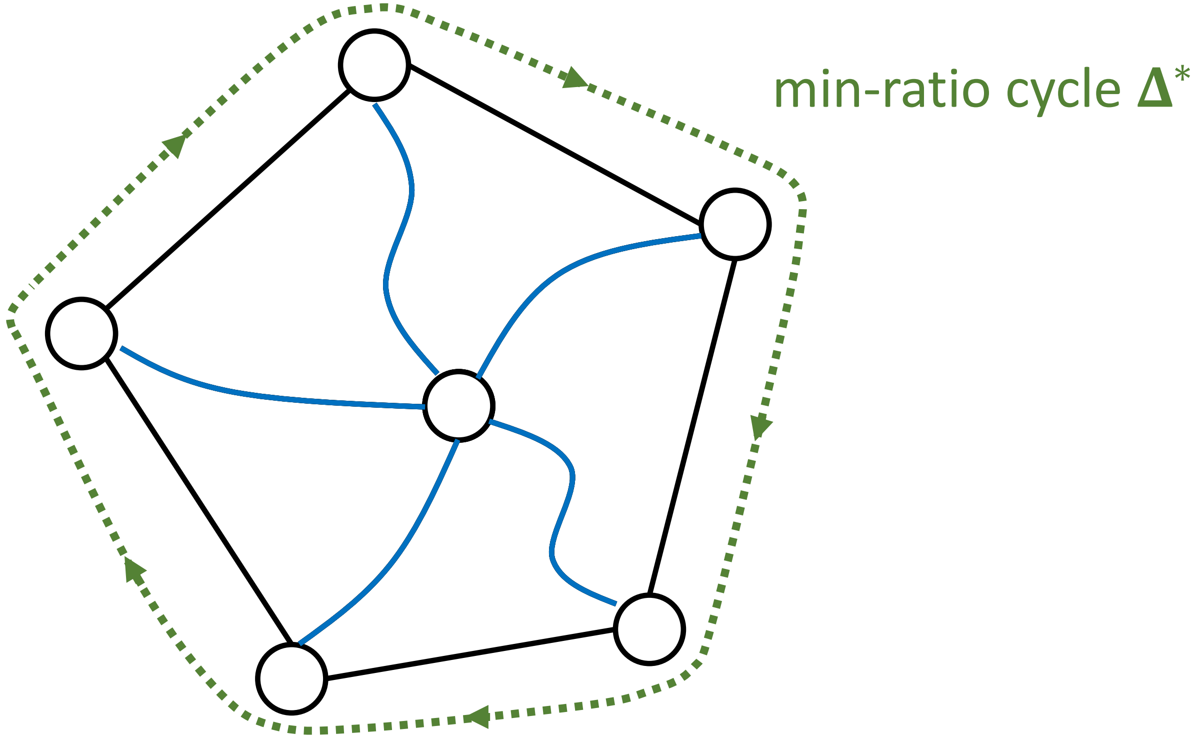
$$\mathbb{E}_T\left[L\left(\mathrm{cycle}_T(e)\right)\right] \leq \tilde{O}(1) L_e$$

Claim: Some $\text{cycle}_T(e)$ is an $\tilde{O}(1)$-approx



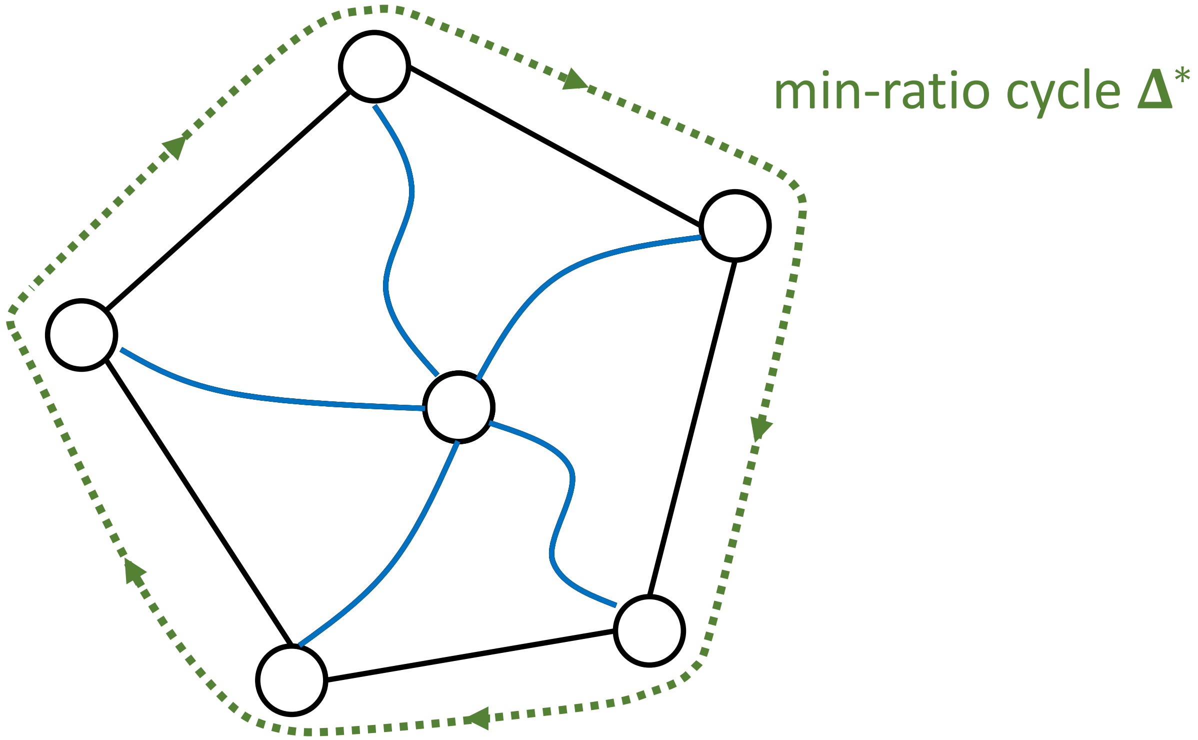
$$\mathbb{E}_T\left[L\left(\text{cycle}_T(e)\right)\right] \leq \tilde{O}(1) L_e$$

Claim: Some $\mathrm{cycle}_T(e)$ is an $\tilde{O}(1)$-approx



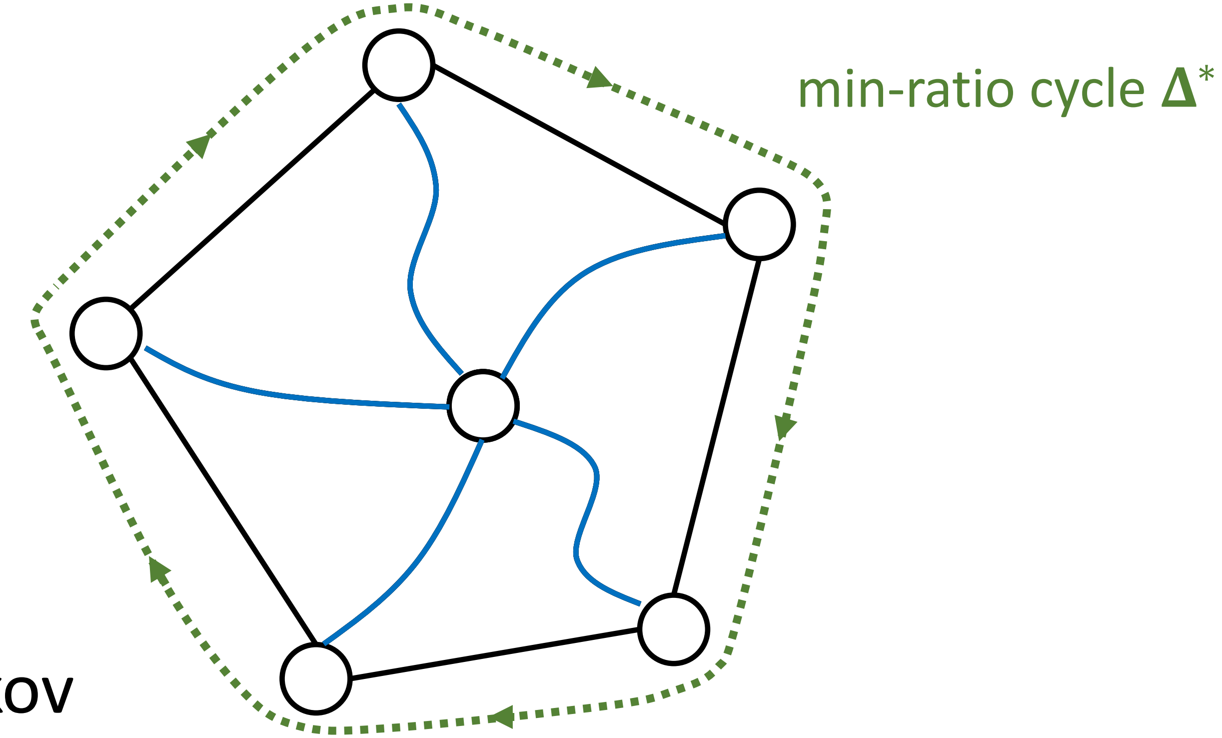
$$\sum_{e \in \Delta^\star} \mathbb{E}_T\left[L\left(\mathrm{cycle}_T(e)\right)\right] \leq \tilde{O}(1) \cdot \|L\Delta^\star\|_1$$

Claim: Some $\mathrm{cycle}_T(e)$ is an $\tilde{O}(1)$-approx



$$\mathbb{E}_T\left[\sum_{e\in\Delta^\star} L\big(\mathrm{cycle}_T(e)\big)\right] \le \tilde{O}(1)\cdot \|L\Delta^\star\|_1$$

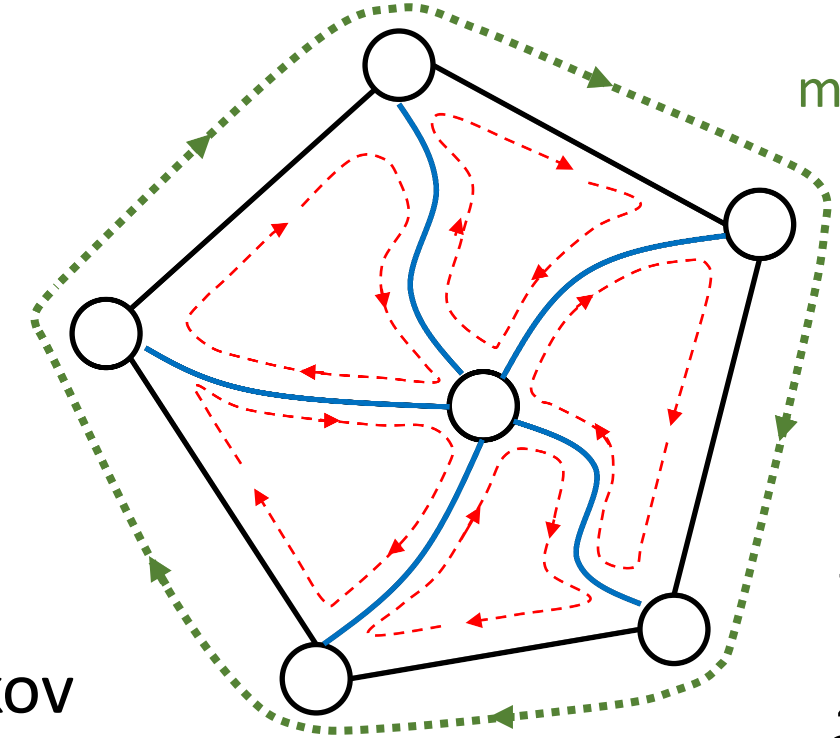
Claim: Some $\text{cycle}_T(e)$ is an $\tilde{O}(1)$-approx



With prob ½, by Markov

$$\sum_{e \in \Delta^\star} L\left(\text{cycle}_T(e)\right) \leq \tilde{O}(1) \cdot \|L\Delta^\star\|_1$$

Claim: Some $\mathrm{cycle}_T(e)$ is an $\tilde{\mathrm{O}}(1)$-approx

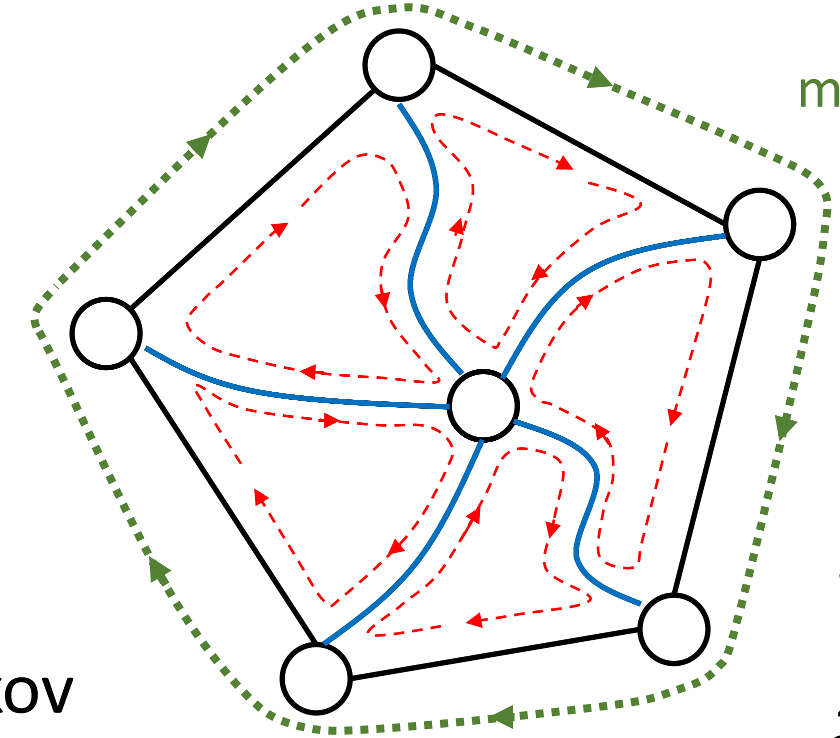


$$\sum_{e \in \Delta^\star} g^\top \mathrm{cycle}_T(e)$$

With prob ½, by Markov

$$\sum_{e \in \Delta^\star} L\big(\mathrm{cycle}_T(e)\big) \le \tilde{O}(1) \cdot \|L\Delta^\star\|_1$$

Claim: Some $\text{cycle}_T(e)$ is an $\tilde{O}(1)$-approx



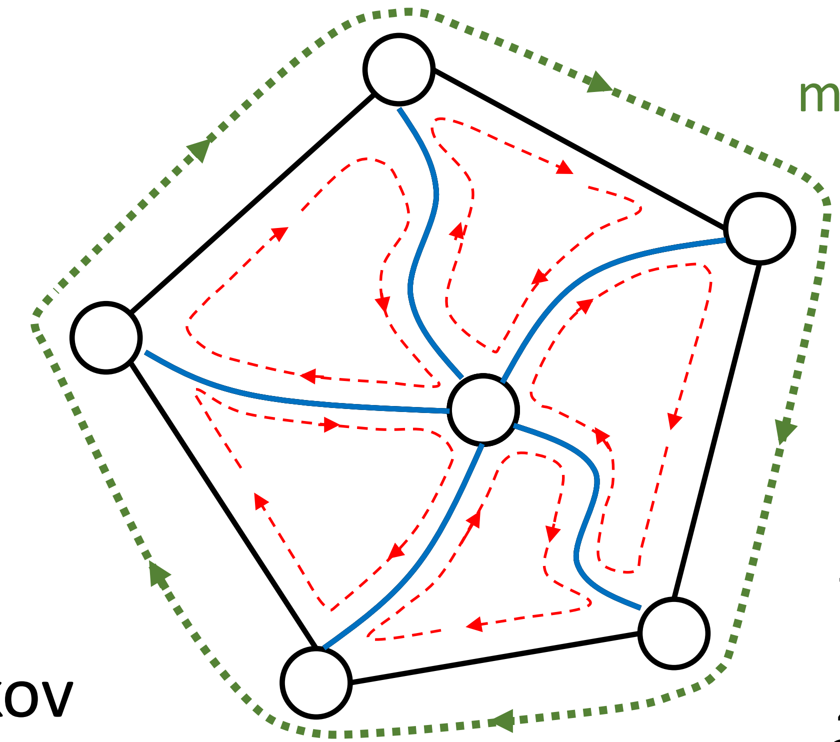
With prob ½, by Markov

$$\sum_{e \in \Delta^\star} L(\text{cycle}_T(e)) \leq \tilde{O}(1) \cdot \|L\Delta^\star\|_1$$

$$\sum_{e \in \Delta^\star} g^\top \text{cycle}_T(e)$$

$$= g^\top \sum_{e \in \Delta^\star} \text{cycle}_T(e)$$

Claim: Some $\text{cycle}_T(e)$ is an $\tilde{O}(1)$-approx



With prob ½, by Markov

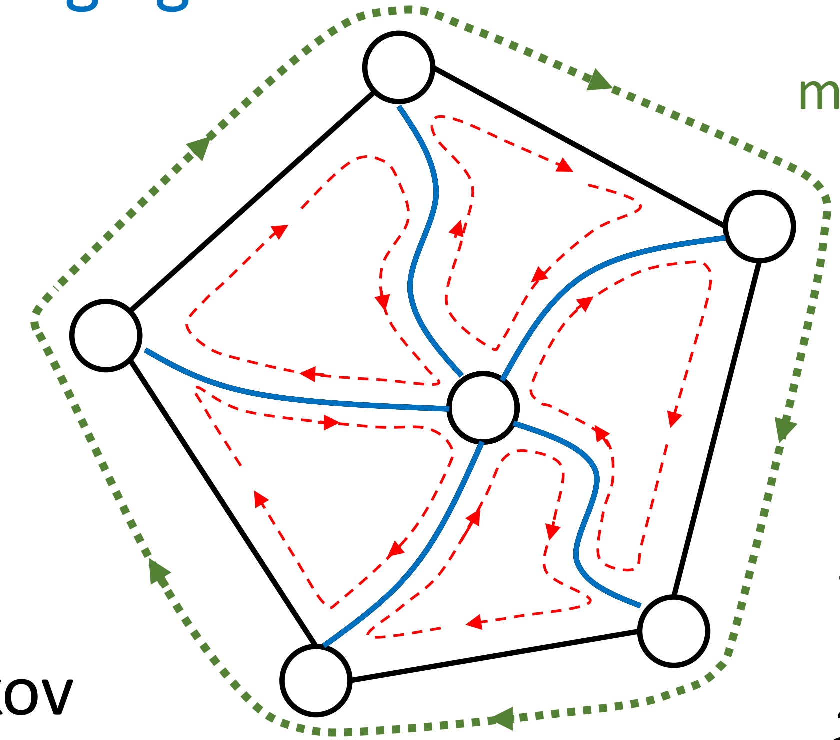
$$\sum_{e \in \Delta^\star} L(\text{cycle}_T(e)) \le \tilde{O}(1) \cdot \|L\Delta^\star\|_1$$

$$\sum_{e \in \Delta^\star} g^\top \text{cycle}_T(e)$$

$$= g^\top \sum_{e \in \Delta^\star} \text{cycle}_T(e) = g^\top \Delta^\star$$

Claim: Some $\text{cycle}_T(e)$ is an $\tilde{O}(1)$-approx

Claim follows by averaging.



With prob ½, by Markov

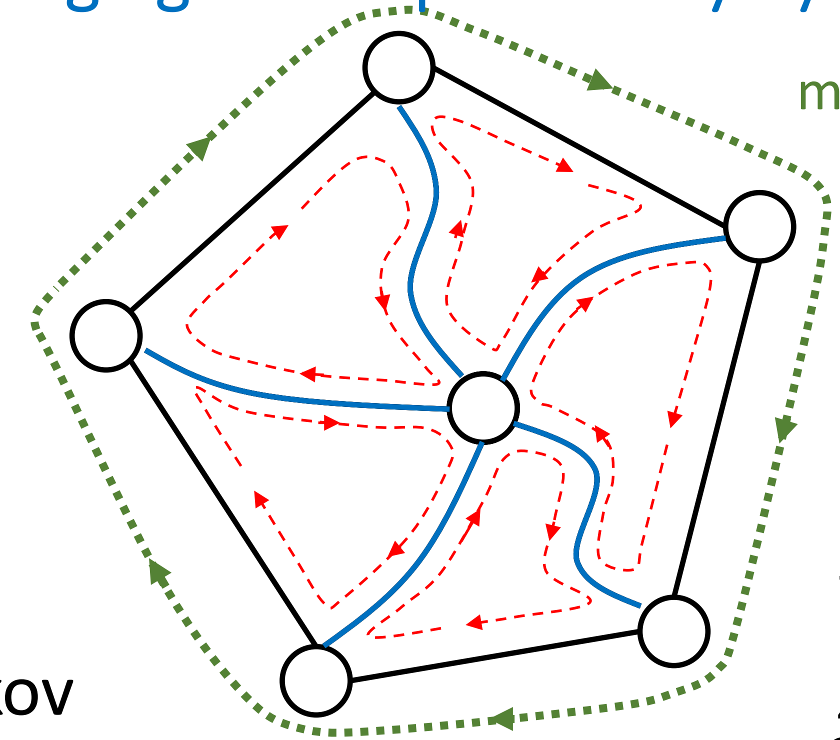
$$\sum_{e\in\Delta^\star} L(\text{cycle}_T(e)) \le \tilde{O}(1)\cdot \|L\Delta^\star\|_1$$

$$\sum_{e\in\Delta^\star} g^\top \text{cycle}_T(e)$$

$$= g^\top \sum_{e\in\Delta^\star} \text{cycle}_T(e) = g^\top \Delta^\star$$

Claim: Some $\text{cycle}_T(e)$ is an $\tilde{O}(1)$-approx

Claim follows by averaging. Boost probability by sampling many trees



With prob ½, by Markov

$$\sum_{e \in \Delta^\star} L(\text{cycle}_T(e)) \le \tilde{O}(1) \cdot \|L\Delta^\star\|_1$$

$$\sum_{e \in \Delta^\star} g^\top \text{cycle}_T(e)$$

$$= g^\top \sum_{e \in \Delta^\star} \text{cycle}_T(e) = g^\top \Delta^\star$$

Min-ratio cycle data-structure

[C-Kyng-L-Peng-Probst Gutenberg-Sachdeva]
A randomized data-structure that maintains $m^{o(1)}$ “low-stretch” trees
And supports in $m^{o(1)}$ amortized time w.h.p.

1. Update g_e, L_e for an edge e
2. Return a $m^{o(1)}$-approximate min-ratio cycle
3. Route flow along such a cycle

Overall Algorithm

A data-structure maintains a few trees $\{T_i\}$

For $t \leftarrow 1, \ldots, m^{1+o(1)}$ iterations

Update gradients g_e and lengths L_e

Update trees $\{T_i\}$ according to L

Identify a circulation Δ approximately minimizing $\frac{g^\top \Delta}{\|L\Delta\|_1}$,

$f^{(t)} \leftarrow f^{(t-1)} + \alpha\Delta$

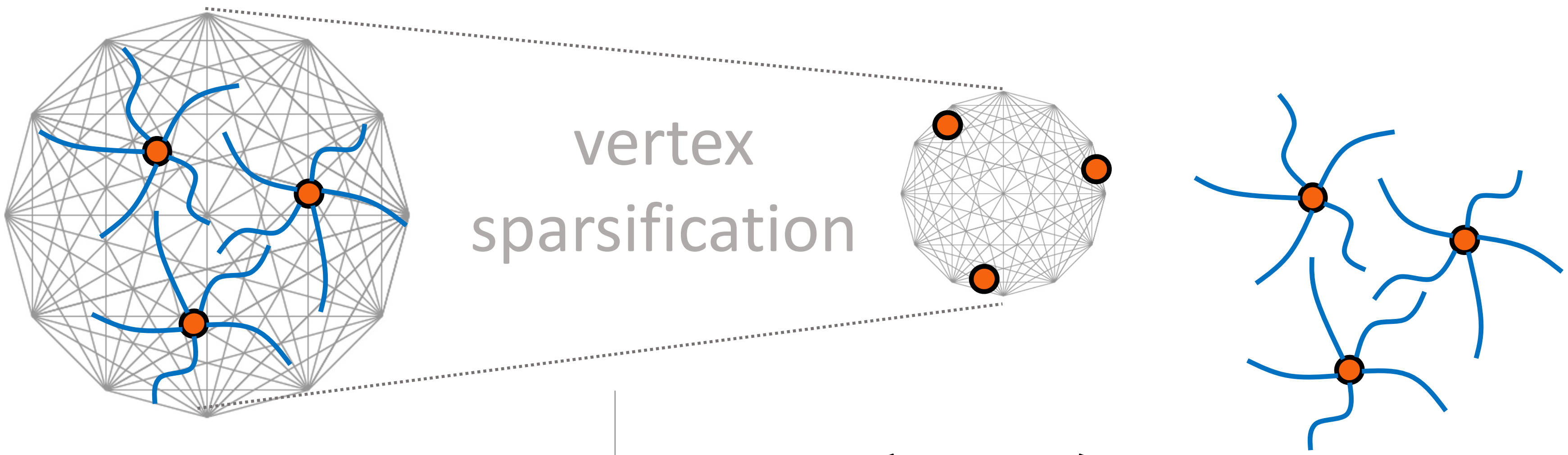
Output final flow $f^{(final)}$

Dynamic Min Ratio Cycle

- "Partial Tree Building"
  - Partial Tree on a subset of vertices/edges $\sim 0.99\ m$
  - Recurse on the rest $\sim 0.01\ m$
- "Partial Tree Maintenance"
  - Maintain partial tree through $0.01\ m$ updates, then rebuild
  - Pass edge update to the recursive DS on the next level
- Challenges:
  - Recursion should reduce #vertices and #edges
  - Maintain the smaller graph under edge updates/vertex splits

Dynamic Min Ratio Cycle

$K = m^{1/d}$



Rooted Forest F
≈"partial tree"

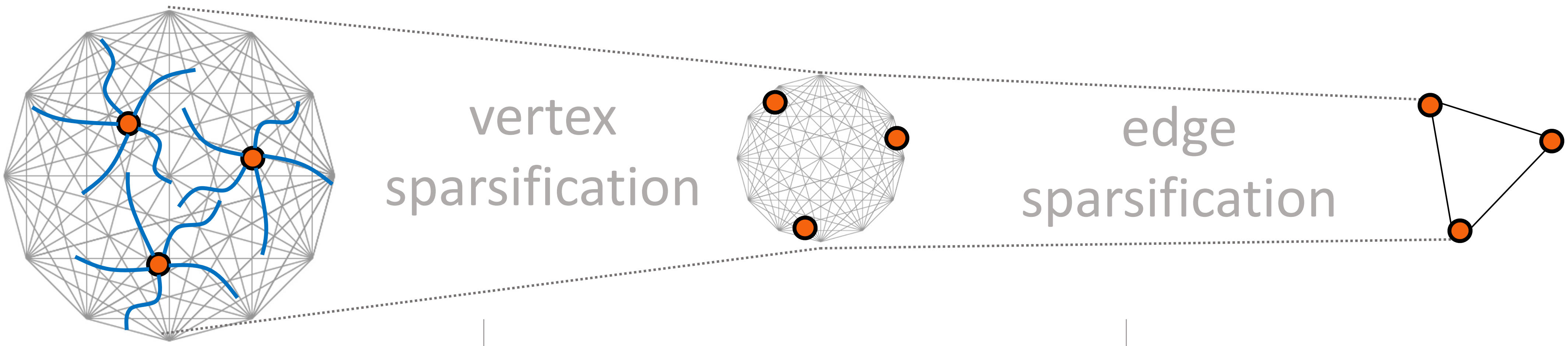
$G = G_0$
≈ m edges
≈ m vertices

$C(G_0, F)$
≈ m edges
≈ m/K vertices
"core graph"

"roots"
=vertices of G_1

Dynamic Min Ratio Cycle

$K = m^{1/d}$



$G = G_0$

$\approx m$ edges

$\approx m$ vertices

$C(G_0, F)$

$\approx m$ edges

$\approx m/K$ vertices

"core graph"

$S(C(G_0, F)) = G_1$

$\approx m/K$ edges

$\approx m/K$ vertices

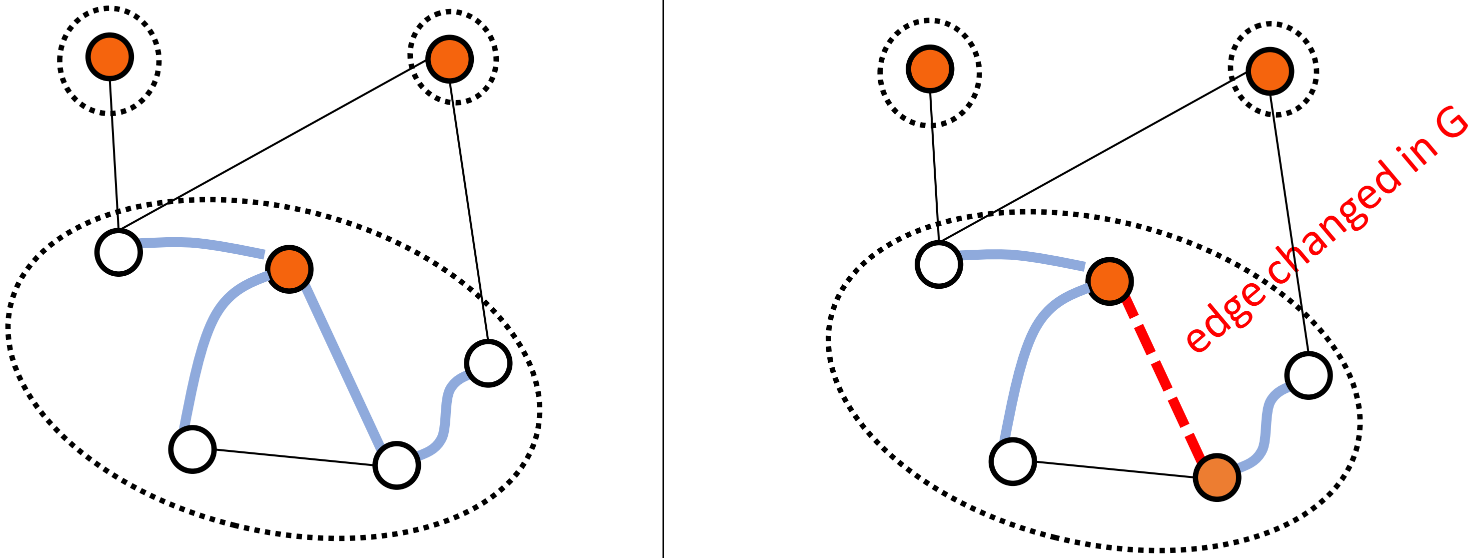
"sparsified core graph"

How $C(G, F)$ Changes?

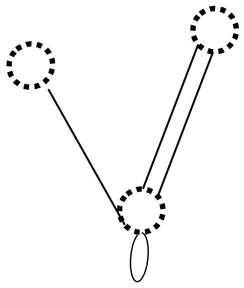
Graph G

Forest F

Roots



Core graph $C(G, F)$

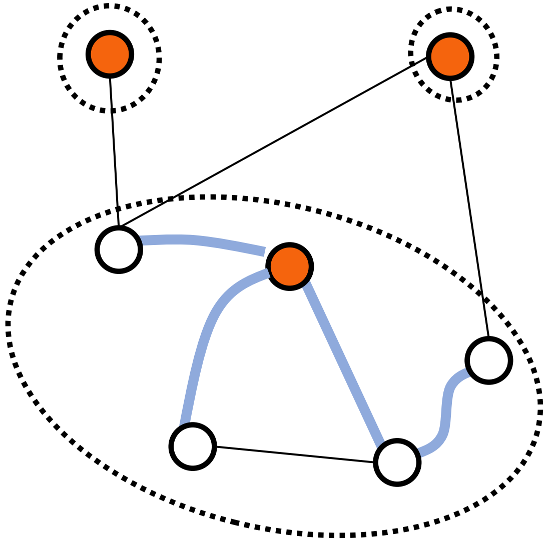


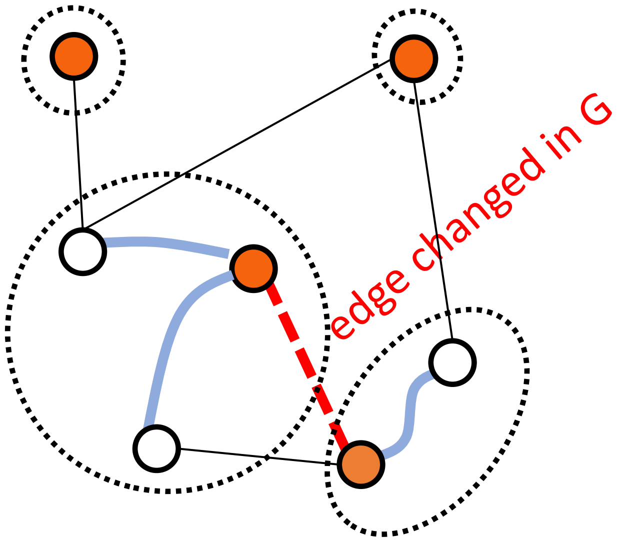
How $C(G, F)$ Changes?

Graph G

Forest F

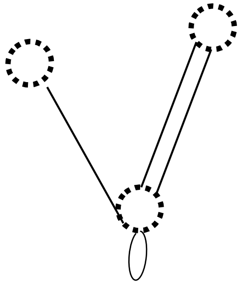
Roots

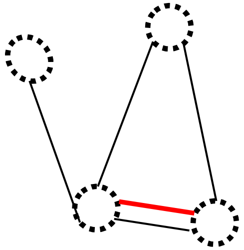




remove it from forest!

Core graph $C(G, F)$





Vertex Split

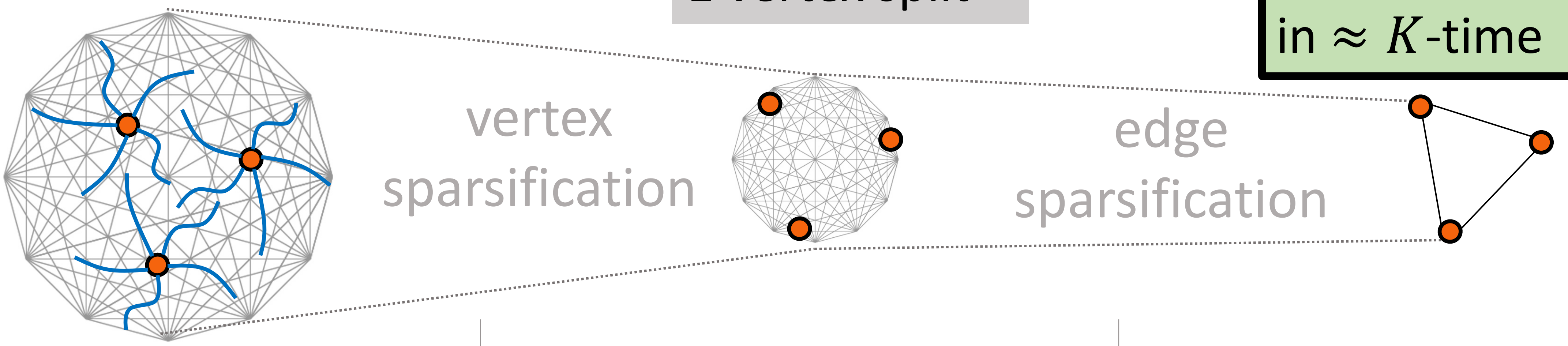
Dynamic Min Ratio Cycle

$K = m^{1/d}$

1 edge update

1 vertex split

$m^{o(1)}$ edge changes in $\approx K$-time



$G = G_0$

$\approx m$ edges

$\approx m$ vertices

$C(G_0, F)$

$\approx m$ edges

$\approx m/K$ vertices

"core graph"

$S(C(G_0, F)) = G_1$

$\approx m/K$ edges

$\approx m/K$ vertices

"sparsified core graph"

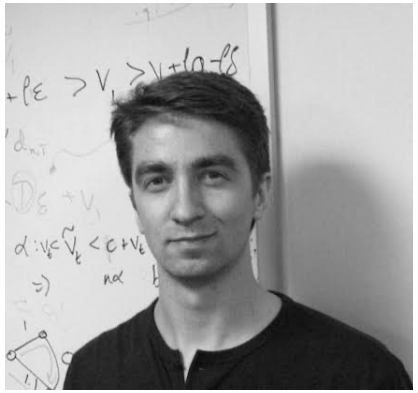
Adaptivity Issue

- Our DS ***does not work*** for all update/query sequences
- gradients g_e and lengths L_e affected by DS output
  non-oblivious adversary
- White-box analysis of DS and IPM
- Gradient/lengths updates reveal which edge becomes important
- $f^* - f$ is a good enough direction

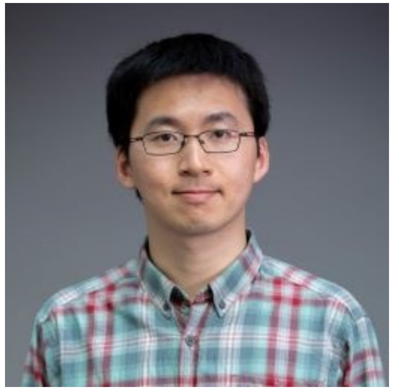
Some immediate open problems

- Deterministic?
- $m^{1+o(1)}$-time to $m\ polylog(m)$–time?
- Static Spanner with Embedding
  Find a sparse subgraph H and
  Embed each edge (u, v) with a $polylog(n)$-length path in H?
- Can we improve k-commodity flow?
- General Graph Matching in Almost-Linear Time?

Thanks!!



Rasmus Kyng
ETH



Richard Peng
U. Waterloo

Maximilian
Probst Gutenberg
ETH

Sushant
Sachdeva
U. Toronto