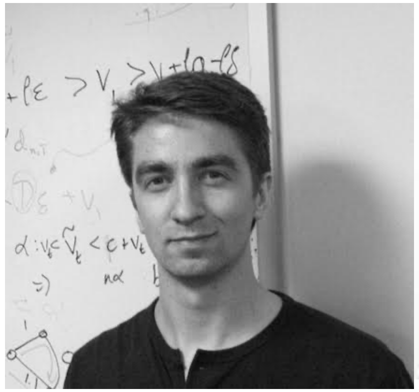
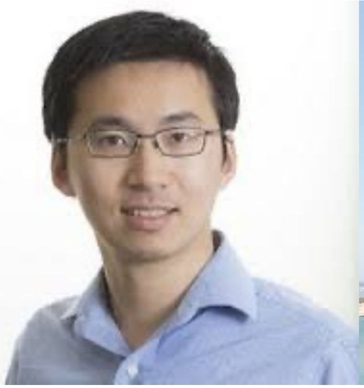
Maximum Flow and Minimum-Cost Flow in Almost Linear Time

Li Chen (Georgia Tech), **Yang P. Liu** (Stanford University)
Joint with Rasmus Kyng, Richard Peng, Maximilian Probst Gutenberg, Sushant Sachdeva

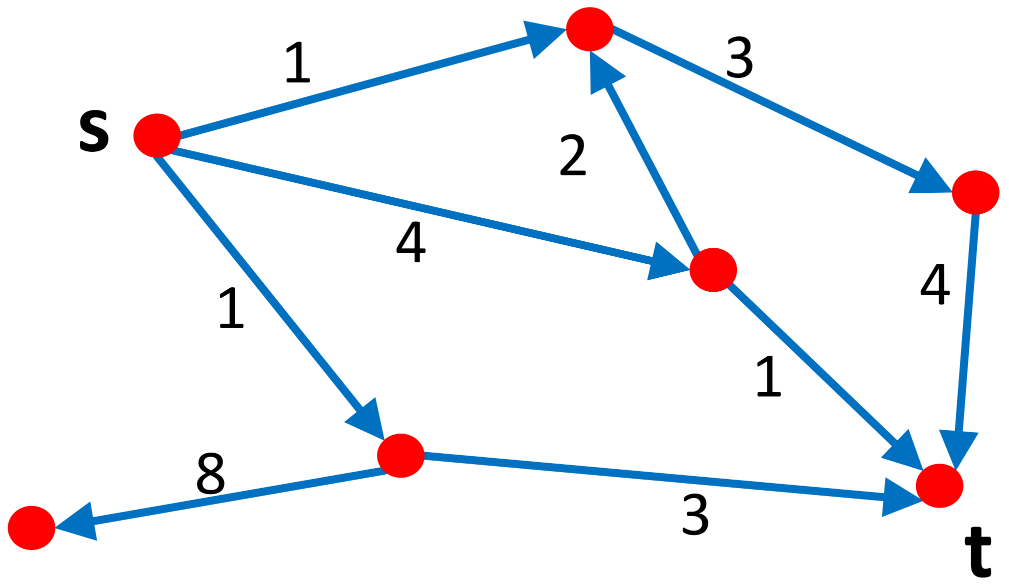
Talk Outline

Part 1: Problem History and Our Results

Part 2: Using Tree Embeddings to Find Approx. Min-Ratio Cycles

Maximum Flow Problem

- Directed graph $G = (V, E)$ source s, sink t, *capacities* $u_e \geq 0$.
- m edges, n vertices, maximum capacity U.

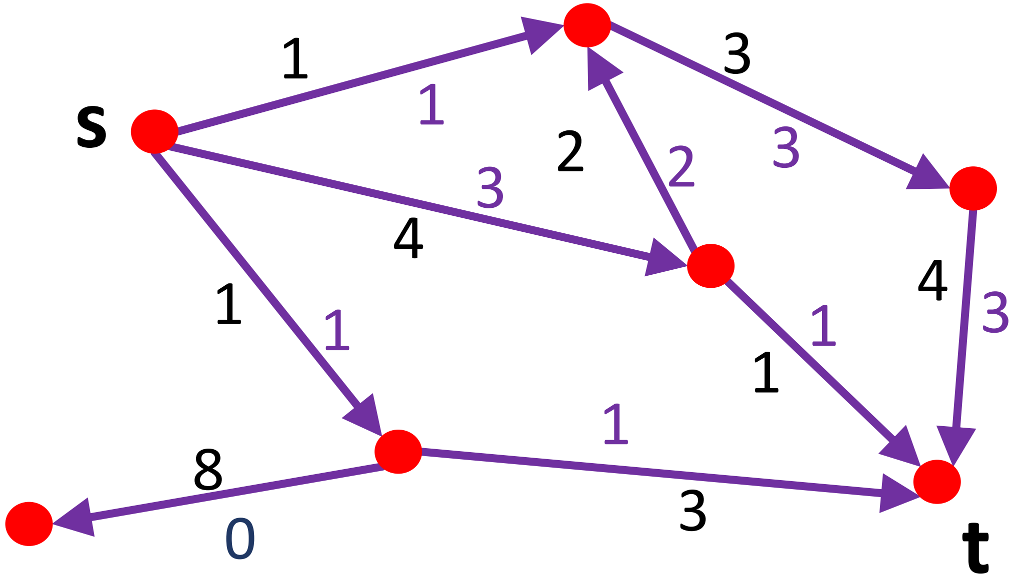


Maximum Flow Problem

- Directed graph G = (V, E) source s, sink t, *capacities* $u_e \geq 0$.
- m edges, n vertices, maximum capacity U.

Goal: Route maximum flow from s -> t.

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

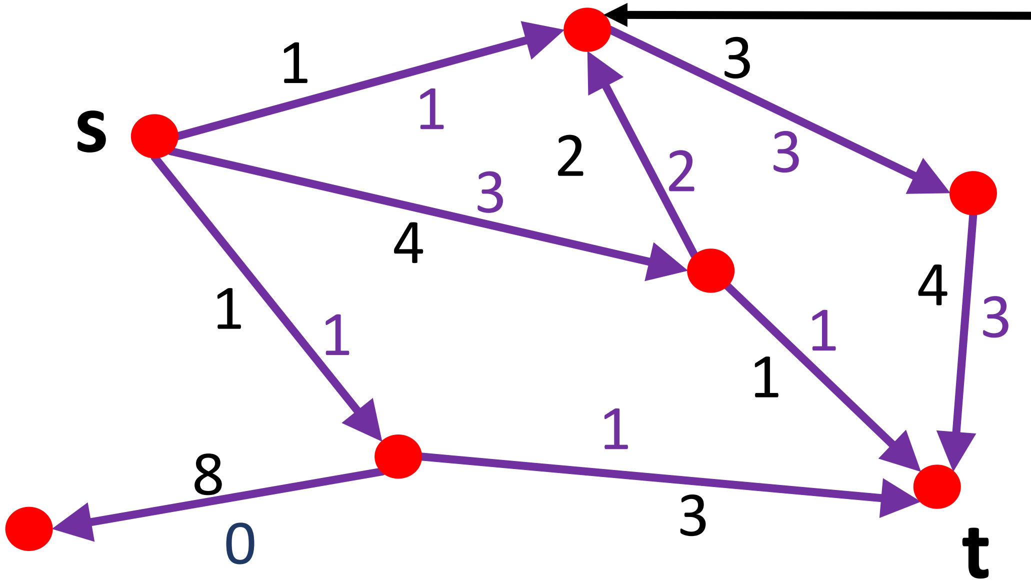


Maximum Flow Problem

- Directed graph G = (V, E) source s, sink t, *capacities* $u_e \geq 0$.
- m edges, n vertices, maximum capacity U.

Goal: Route maximum flow from s -> t.

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



Demand constraint: all vertices except s, t have equal incoming/outgoing flow

Total flow: number of units leaving s / entering t

Capacity constraint: amount of flow on edge e in [0, u_e]

Why Study Flows?

- **Graph opt:** Flows are a broad class of graph optimization problems.
- Route 1 unit from s to t while minimizing some *cost* given by convex functions on edges, $\sum_{\mathrm{edges}\ e} \mathrm{cost}_{\mathrm{e}}(f_{\mathrm{e}})$.
- Covers minimum-cost flow, optimal transport, isotonic regression, p-norm flows, regularized optimal transport, matrix scaling, etc.

Why Study Flows?

- **Graph opt:** Flows are a broad class of graph optimization problems.
- Route 1 unit from s to t while minimizing some *cost* given by convex functions on edges, $\sum_{\mathrm{edges}\ e} \mathrm{cost}_{\mathrm{e}}(f_{\mathrm{e}})$.
- Covers minimum-cost flow, optimal transport, isotonic regression, p-norm flows, regularized optimal transport, matrix scaling, etc.

Our results cover all the problems mentioned above.

- **Direct applications:** Bipartite matching, densest subgraph, Gomory-Hu trees / connectivity, negative-weight shortest path

$\tilde{O}(.)$ hides poly(log(mnU)) factors

Comparison of Previous Algorithms

	Runtime	Iterations	Amortized Cost
[GN80, ST83]	$O(mn \log n)$	$\tilde{O}(n)$	$\tilde{O}(m)$
[GR98]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[DS08]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[M13, M16]	$\tilde{O}(m^{10/7}U^{1/7})$	$\tilde{O}(m^{3/7}U^{1/7})$	$\tilde{O}(m)$
[K**L**S20]	$m^{4/3+o(1)}U^{1/3}$	$m^{1/3+o(1)}U^{1/3}$	$m^{1+o(1)}$
[LS14]	$\tilde{O}(mn^{1/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m)$
[BLNPSSSW20, BL**L**SSSW21]	$\tilde{O}(m + n^{3/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m/n^{1/2} + n)$
[G**L**P21, BGJL**L**PS22]	$\tilde{O}(m^{3/2-1/58})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m^{1-1/58})$

$\tilde{O}(.)$ hides poly(log(mnU)) factors

Comparison of Previous Algorithms

	Runtime	Iterations	Amortized Cost
[GN80, ST83]	$O(mn \log n)$	$\tilde{O}(n)$	$\tilde{O}(m)$
[GR98]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[DS08]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[M13, M16]	$\tilde{O}(m^{10/7}U^{1/7})$	$\tilde{O}(m^{3/7}U^{1/7})$	$\tilde{O}(m)$
[K**L**S20]	$m^{4/3+o(1)}U^{1/3}$	$m^{1/3+o(1)}U^{1/3}$	$m^{1+o(1)}$
[LS14]	$\tilde{O}(mn^{1/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m)$
[BLNPSSSW20, BL**L**SSSW21]	$\tilde{O}(m + n^{3/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m/n^{1/2} + n)$
[G**L**P21, BGJL**L**PS22]	$\tilde{O}(m^{3/2-1/58})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m^{1-1/58})$
[CK**L**PPS22]	$m^{1+o(1)}$	$m^{1+o(1)}$	$m^{o(1)}$

Õ(.) hides poly(log(mnU)) factors

Comparison of Previous Algorithms

	Runtime	Iterations	Amortized Cost
[GN80, ST83]	$O(mn \log n)$	$\tilde{O}(n)$	$\tilde{O}(m)$
[GR98]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[DS08]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[M13, M16]	$\tilde{O}(m^{10/7}U^{1/7})$	$\tilde{O}(m^{3/7}U^{1/7})$	$\tilde{O}(m)$
[K**L**S20]	$m^{4/3+o(1)}U^{1/3}$	$m^{1/3+o(1)}U^{1/3}$	$m^{1+o(1)}$
[LS14]	$\tilde{O}(mn^{1/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m)$
[BLNPSSSW20, BL**L**SSSW21]	$\tilde{O}(m + n^{3/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m/n^{1/2} + n)$
[G**L**P21, BGJL**L**PS22]	$\tilde{O}(m^{3/2-1/58})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m^{1-1/58})$
[CK**L**PPS22]	$m^{1+o(1)}$	$m^{1+o(1)}$	$m^{o(1)}$

Augmenting paths / blocking flows and dynamic tree data structure.

$\tilde{O}(.)$ hides poly(log(mnU)) factors

Comparison of Previous Algorithms

	Runtime	Iterations	Amortized Cost
[GN80, ST83]	$O(mn \log n)$	$\tilde{O}(n)$	$\tilde{O}(m)$
[GR98]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[DS08]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[M13, M16]	$\tilde{O}(m^{10/7}U^{1/7})$	$\tilde{O}(m^{3/7}U^{1/7})$	$\tilde{O}(m)$
[K**L**S20]	$m^{4/3+o(1)}U^{1/3}$	$m^{1/3+o(1)}U^{1/3}$	$m^{1+o(1)}$
[LS14]	$\tilde{O}(mn^{1/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m)$
[BLNPSSSW20, BL**L**SSSW21]	$\tilde{O}(m + n^{3/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m/n^{1/2} + n)$
[G**L**P21, BGJL**L**PS22]	$\tilde{O}(m^{3/2-1/58})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m^{1-1/58})$
[CK**L**PPS22]	$m^{1+o(1)}$	$m^{1+o(1)}$	$m^{o(1)}$

Blocking flows, capacity scaling, and much more.

Õ(.) hides poly(log(mnU)) factors

Comparison of Previous Algorithms

	Runtime	Iterations	Amortized Cost
[GN80, ST83]	$O(mn \log n)$	$\tilde{O}(n)$	$\tilde{O}(m)$
[GR98]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[DS08]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[M13, M16]	$\tilde{O}(m^{10/7}U^{1/7})$	$\tilde{O}(m^{3/7}U^{1/7})$	$\tilde{O}(m)$
[K**L**S20]	$m^{4/3+o(1)}U^{1/3}$	$m^{1/3+o(1)}U^{1/3}$	$m^{1+o(1)}$
[LS14]	$\tilde{O}(mn^{1/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m)$
[BLNPSSSW20, BL**L**SSSW21]	$\tilde{O}(m + n^{3/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m/n^{1/2} + n)$
[G**L**P21, BGJL**L**PS22]	$\tilde{O}(m^{3/2-1/58})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m^{1-1/58})$
[CK**L**PPS22]	$m^{1+o(1)}$	$m^{1+o(1)}$	$m^{o(1)}$

Interior point method (IPM) + Laplacian solver: also min-cost.

$\tilde{O}(.)$ hides poly(log(mnU)) factors

Comparison of Previous Algorithms

	Runtime	Iterations	Amortized Cost
[GN80, ST83]	$O(mn \log n)$	$\tilde{O}(n)$	$\tilde{O}(m)$
[GR98]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[DS08]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[M13, M16]	$\tilde{O}(m^{10/7}U^{1/7})$	$\tilde{O}(m^{3/7}U^{1/7})$	$\tilde{O}(m)$
[K**L**S20]	$m^{4/3+o(1)}U^{1/3}$	$m^{1/3+o(1)}U^{1/3}$	$m^{1+o(1)}$
[LS14]	$\tilde{O}(mn^{1/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m)$
[BLNPSSSW20, BL**L**SSSW21]	$\tilde{O}(m + n^{3/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m/n^{1/2} + n)$
[G**L**P21, BGJL**L**PS22]	$\tilde{O}(m^{3/2-1/58})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m^{1-1/58})$
[CK**L**PPS22]	$m^{1+o(1)}$	$m^{1+o(1)}$	$m^{o(1)}$

IPM + Laplacian + weight changes to reduce iterations.

$\tilde{O}(.)$ hides poly(log(mnU)) factors

Comparison of Previous Algorithms

	Runtime	Iterations	Amortized Cost
[GN80, ST83]	$O(mn \log n)$	$\tilde{O}(n)$	$\tilde{O}(m)$
[GR98]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[DS08]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[M13, M16]	$\tilde{O}(m^{10/7}U^{1/7})$	$\tilde{O}(m^{3/7}U^{1/7})$	$\tilde{O}(m)$
[K**L**S20]	$m^{4/3+o(1)}U^{1/3}$	$m^{1/3+o(1)}U^{1/3}$	$m^{1+o(1)}$
[LS14]	$\tilde{O}(mn^{1/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m)$
[BLNPSSSW20, BL**L**SSSW21]	$\tilde{O}(m + n^{3/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m/n^{1/2} + n)$
[G**L**P21, BGJL**L**PS22]	$\tilde{O}(m^{3/2-1/58})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m^{1-1/58})$
[CK**L**PPS22]	$m^{1+o(1)}$	$m^{1+o(1)}$	$m^{o(1)}$

IPM + p-norm flows: more sophisticated iteration reduction

$\tilde{O}(.)$ hides poly(log(mnU)) factors

Comparison of Previous Algorithms

	Runtime	**Iterations**	**Amortized Cost**
[GN80, ST83]	$O(mn \log n)$	$\tilde{O}(n)$	$\tilde{O}(m)$
[GR98]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[DS08]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[M13, M16]	$\tilde{O}(m^{10/7}U^{1/7})$	$\tilde{O}(m^{3/7}U^{1/7})$	$\tilde{O}(m)$
[K**L**S20]	$m^{4/3+o(1)}U^{1/3}$	$m^{1/3+o(1)}U^{1/3}$	$m^{1+o(1)}$
[LS14]	$\tilde{O}(mn^{1/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m)$
[BLNPSSSW20, BL**L**SSSW21]	$\tilde{O}(m + n^{3/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m/n^{1/2} + n)$
[G**L**P21, BGJL**L**PS22]	$\tilde{O}(m^{3/2-1/58})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m^{1-1/58})$
[CK**L**PPS22]	$m^{1+o(1)}$	$m^{1+o(1)}$	$m^{o(1)}$

IPM + generic iteration count reduction for all linear programs

$\tilde{O}(.)$ hides poly(log(mnU)) factors

Comparison of Previous Algorithms

	Runtime	Iterations	Amortized Cost
[GN80, ST83]	$O(mn \log n)$	$\tilde{O}(n)$	$\tilde{O}(m)$
[GR98]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[DS08]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[M13, M16]	$\tilde{O}(m^{10/7}U^{1/7})$	$\tilde{O}(m^{3/7}U^{1/7})$	$\tilde{O}(m)$
[K**L**S20]	$m^{4/3+o(1)}U^{1/3}$	$m^{1/3+o(1)}U^{1/3}$	$m^{1+o(1)}$
[LS14]	$\tilde{O}(mn^{1/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m)$
[BLNPSSSW20, BL**L**SSSW21]	$\tilde{O}(m + n^{3/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m/n^{1/2} + n)$
[G**L**P21, BGJL**L**PS22]	$\tilde{O}(m^{3/2-1/58})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m^{1-1/58})$
[CK**L**PPS22]	$m^{1+o(1)}$	$m^{1+o(1)}$	$m^{o(1)}$

IPM + dynamic electrical flows: heavy-hitters, sparsification

$\tilde{O}(.)$ hides poly(log(mnU)) factors

Comparison of Previous Algorithms

	Runtime	Iterations	Amortized Cost
[GN80, ST83]	$O(mn \log n)$	$\tilde{O}(n)$	$\tilde{O}(m)$
[GR98]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[DS08]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[M13, M16]	$\tilde{O}(m^{10/7}U^{1/7})$	$\tilde{O}(m^{3/7}U^{1/7})$	$\tilde{O}(m)$
[K**L**S20]	$m^{4/3+o(1)}U^{1/3}$	$m^{1/3+o(1)}U^{1/3}$	$m^{1+o(1)}$
[LS14]	$\tilde{O}(mn^{1/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m)$
[BLNPSSSW20, BL**L**SSSW21]	$\tilde{O}(m + n^{3/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m/n^{1/2} + n)$
[G**L**P21, BGJL**L**PS22]	$\tilde{O}(m^{3/2-1/58})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m^{1-1/58})$
[CK**L**PPS22]	$m^{1+o(1)}$	$m^{1+o(1)}$	$m^{o(1)}$

IPM + dynamic electrical flows via random walk + Schur complement

$\tilde{O}(.)$ hides poly(log(mnU)) factors

Comparison of Previous Algorithms

	Runtime	Iterations	Amortized Cost
[GN80, ST83]	$O(mn \log n)$	$\tilde{O}(n)$	$\tilde{O}(m)$
[GR98]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[DS08]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[M13, M16]	$\tilde{O}(m^{10/7}U^{1/7})$	$\tilde{O}(m^{3/7}U^{1/7})$	$\tilde{O}(m)$
[K**L**S20]	$m^{4/3+o(1)}U^{1/3}$	$m^{1/3+o(1)}U^{1/3}$	$m^{1+o(1)}$
[LS14]	$\tilde{O}(mn^{1/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m)$
[BLNPSSSW20, BL**L**SSSW21]	$\tilde{O}(m + n^{3/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m/n^{1/2} + n)$
[G**L**P21, BGJL**L**PS22]	$\tilde{O}(m^{3/2-1/58})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m^{1-1/58})$
[CK**L**PPS22]	$m^{1+o(1)}$	$m^{1+o(1)}$	$m^{o(1)}$

IPM + dynamic approx. minimum-ratio *cycles*, **not electric flows**

Õ(.) hides poly(log(mnU)) factors

Comparison of Previous Algorithms

$\tilde{O}(\boldsymbol{m}/\boldsymbol{\epsilon})$ for $(1+\epsilon)$-approximate *undirected* max-flow [S13,KLOS14,P16,BGS22]

	Runtime	Iterations	Amortized Cost
[GN80, ST83]	$O(mn \log n)$	$\tilde{O}(n)$	$\tilde{O}(m)$
[GR98]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[DS08]	$\tilde{O}(m^{3/2})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m)$
[M13, M16]	$\tilde{O}(m^{10/7}U^{1/7})$	$\tilde{O}(m^{3/7}U^{1/7})$	$\tilde{O}(m)$
[KLS20]	$m^{4/3+o(1)}U^{1/3}$	$m^{1/3+o(1)}U^{1/3}$	$m^{1+o(1)}$
[LS14]	$\tilde{O}(mn^{1/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m)$
[BLNPSSSW20, BLLSSSW21]	$\tilde{O}(m + n^{3/2})$	$\tilde{O}(n^{1/2})$	$\tilde{O}(m/n^{1/2} + n)$
[GLP21, BGJLLPS22]	$\tilde{O}(m^{3/2-1/58})$	$\tilde{O}(m^{1/2})$	$\tilde{O}(m^{1-1/58})$
[CKLPPS22]	$m^{1+o(1)}$	$m^{1+o(1)}$	$m^{o(1)}$

IPM + dynamic approx. minimum-ratio *cycles*, **not electric flows**

Intuition for Improvements

- **Combinatorial algorithms** [GN80, ST83, GT87]
  - Augmenting paths, shortest paths, blocking flows.
  - Cycle cancelling by finding *minimum-mean cycles.*
  - Work primarily with residual graphs that are *directed*.
- **Continuous optimization** [DS08, M13/16, LS14, etc.]
  - Augmenting *electrical flows* (ℓ_2 primitive) or *circulations*
  - IPMs allow for the electrical flows to be *undirected*.
  - Only $\tilde{O}(m^{1/2})$ iterations, while a flow requires $\Omega(m)$ paths.

Intuition for Improvements

- **Combinatorial algorithms** [GN80, ST83, GT87]
  - Augmenting paths, shortest paths, blocking flows.
  - Cycle cancelling by finding *minimum-mean cycles.*
  - Work primarily with residual graphs that are *directed*.
- **Continuous optimization** [DS08, M13/16, LS14, etc.]
  - Augmenting *electrical flows* (ℓ_2 primitive) or *circulations*
  - IPMs allow for the electrical flows to be *undirected*.
  - Only $\tilde{O}(m^{1/2})$ iterations, while a flow requires $\Omega(m)$ paths.
- **Our algorithm:**
  - Approx. minimum-ratio *cycles* to build flow: $m^{1+o(1)}$ iters.
  - Computing min-ratio cycles is an *undirected* flow problem.

Intuition for Improvements

- **Combinatorial algorithms** [GN80, ST83, GT87]
  - Augmenting paths, shortest paths, blocking flows.
  - Cycle cancelling by finding *minimum-mean cycles.*
  - Work primarily with residual graphs that are *directed*.
- **Continuous optimization** [DS08, M13/16, LS14, etc.]
  - Augmenting *electrical flows* (ℓ_2 primitive) or *circulations*
  - IPMs allow for the electrical flows to be *undirected*.
  - Only $\tilde{O}(m^{1/2})$ iterations, while a flow requires $\Omega(m)$ paths.
- **Our algorithm:**
  - Approx. minimum-ratio *cycles* to build flow: $m^{1+o(1)}$ iters.
  - Computing min-ratio cycles is an *undirected* flow problem.

Minimize over cycles C in a directed graph: $cost(C)/len(C)$,

Intuition for Improvements

- **Combinatorial algorithms** [GN80, ST83, GT87]
  - Augmenting paths, shortest paths, blocking flows.
  - Cycle cancelling by finding *minimum-mean cycles.*
  - Work primarily with residual graphs that are *directed*.
- **Continuous optimization** [DS08, M13/16, LS14, etc.]
  - Augmenting *electrical flows* (ℓ_2 primitive) or *circulations*
  - IPMs allow for the electrical flows to be *undirected*.
  - Only Õ($m^{1/2}$) iterations, while a flow requires $\Omega(m)$ paths.
- **Our algorithm:**
  - Approx. minimum-ratio *cycles* to build flow: $m^{1+o(1)}$ iters.
  - Computing min-ratio cycles is an *undirected* flow problem.

Minimize over cycles C in a directed graph: $cost(C)/len(C)$,

Minimize over cycles C in an **undirected** graph: $\langle g, C\rangle/||LC||_2$.

Intuition for Improvements

- **Combinatorial algorithms** [GN80, ST83, GT87]
  - Augmenting paths, shortest paths, blocking flows.
  - Cycle cancelling by finding *minimum-mean cycles.*
  - Work primarily with residual graphs that are *directed*.
- **Continuous optimization** [DS08, M13/16, LS14, etc.]
  - Augmenting *electrical flows* (ℓ_2 primitive) or *circulations*
  - IPMs allow for the electrical flows to be *undirected*.
  - Only Õ($m^{1/2}$) iterations, while a flow requires $\Omega(m)$ paths.
- **Our algorithm:**
  - Approx. minimum-ratio *cycles* to build flow: $m^{1+o(1)}$ iters.
  - Computing min-ratio cycles is an *undirected* flow problem.

Minimize over cycles C in a directed graph: $cost(C)/len(C)$,

Minimize over cycles C in an **undirected** graph: $\langle g, C\rangle / ||LC||_2$.

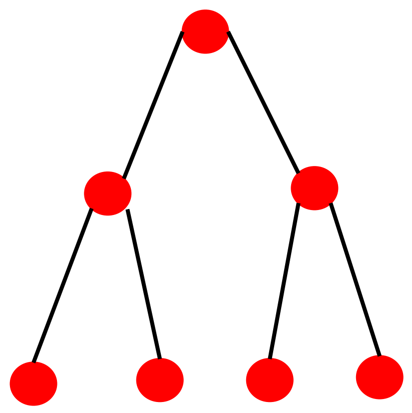
Minimize over cycles C in an **undirected** graph: $\langle g, C\rangle / ||LC||_1$.

Optimization Method Result: Reduction to Slowly Changing Min-Ratio Cycle Instances

- (Informal Theorem) We compute a maxflow in $m^{1+o(1)}$ iterations of:
- Add circulation c which is $m^{o(1)}$-approx. minimizer to $\langle g, c\rangle / ||Lc||_1$ over circulations, for dynamically changing gradients g, lengths L.
- Coordinates of g, L change at most $m^{1+o(1)}$ times total $m^{1+o(1)}$

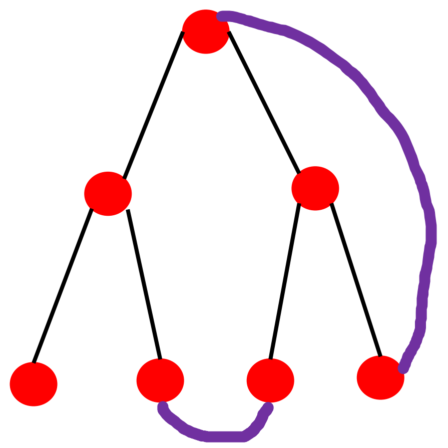
Optimization Method Result: Reduction to Slowly Changing Min-Ratio Cycle Instances

- (Informal Theorem) We compute a maxflow in $m^{1+o(1)}$ iterations of:
- Add circulation c which is $m^{o(1)}$-approx. minimizer to $\langle g, c\rangle / ||Lc||_1$ over circulations, for dynamically changing gradients g, lengths L.
- Coordinates of g, L change at most $m^{1+o(1)}$ times total $m^{1+o(1)}$
- **Implementation:** c will be represented via $m^{o(1)}$ paths on a slowly changing tree T
- So: add c and detect when to change g, L using dynamic tree DS.



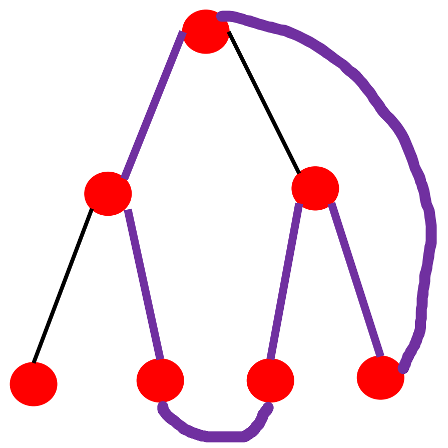
Optimization Method Result: Reduction to Slowly Changing Min-Ratio Cycle Instances

- (Informal Theorem) We compute a maxflow in $m^{1+o(1)}$ iterations of:
- Add circulation c which is $m^{o(1)}$-approx. minimizer to $\langle g, c\rangle / ||Lc||_1$ over circulations, for dynamically changing gradients g, lengths L.
- Coordinates of g, L change at most $m^{1+o(1)}$ times total $m^{1+o(1)}$
- **Implementation:** c will be represented via $m^{o(1)}$ paths on a slowly changing tree T
- So: add c and detect when to change g, L using dynamic tree DS.



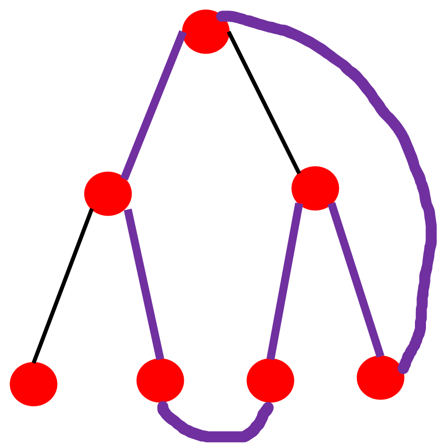
Optimization Method Result: Reduction to Slowly Changing Min-Ratio Cycle Instances

- (Informal Theorem) We compute a maxflow in $m^{1+o(1)}$ iterations of:
- Add circulation c which is $m^{o(1)}$-approx. minimizer to $\langle g, c\rangle / ||Lc||_1$ over circulations, for dynamically changing gradients g, lengths L.
- Coordinates of g, L change at most $m^{1+o(1)}$ times total $m^{1+o(1)}$
- **Implementation:** c will be represented via $m^{o(1)}$ paths on a slowly changing tree T
- So: add c and detect when to change g, L using dynamic tree DS.



Optimization Method Result: Reduction to Slowly Changing Min-Ratio Cycle Instances

- (Informal Theorem) We compute a maxflow in $m^{1+o(1)}$ iterations of:
- Add circulation c which is $m^{o(1)}$-approx. minimizer to $\langle g, c\rangle / ||Lc||_1$ over circulations, for dynamically changing gradients g, lengths L.
- Coordinates of g, L change at most $m^{1+o(1)}$ times total $m^{1+o(1)}$
- **Implementation:** c will be represented via $m^{o(1)}$ paths on a slowly changing tree T
- So: add c and detect when to change g, L using dynamic tree DS.
- Simpler version given in [Wallacher-Zimmerman, Math. Prog. '92]



Dynamically Finding Approx. Min-Ratio Cycles

- (Informal theorem) Randomized data structure that supports:
  - Change gradients/lengths g, L.
  - Return a cycle c that $m^{o(1)}$ approx. minimizes $\langle g, c\rangle / ||Lc||_1$.
  - c is representing as $m^{o(1)}$ paths + off-tree edges on an explicit tree T.
- $m^{o(1)}$ amortized time, works whp. against oblivious adversaries

Dynamically Finding Approx. Min-Ratio Cycles

- (Informal theorem) Randomized data structure that supports:
  - Change gradients/lengths g, L.
  - Return a cycle c that $m^{o(1)}$ approx. minimizes $\langle g, c\rangle / ||Lc||_1$.
  - c is representing as $m^{o(1)}$ paths + off-tree edges on an explicit tree T.
- $m^{o(1)}$ amortized time, works whp. against oblivious adversaries
- (Informal theorem 2) Randomized data structure that supports the above operations, but works for the (possibly non-oblivious) instances generated during the optimization method

Dynamically Finding Approx. Min-Ratio Cycles

- (Informal theorem) Randomized data structure that supports:
  - Change gradients/lengths g, L.
  - Return a cycle c that $m^{o(1)}$ approx. minimizes $\langle g, c\rangle / ||Lc||_1$.
  - c is representing as $m^{o(1)}$ paths + off-tree edges on an explicit tree T.
- $m^{o(1)}$ amortized time, works whp. against oblivious adversaries
- (Informal theorem 2) Randomized data structure that supports the above operations, but works for the (possibly non-oblivious) instances generated during the optimization method

Framework goes beyond the standard oblivious/adaptive split.

Talk Outline

Part 1: Problem History and Our Results

Part 2: Using Tree Embeddings to Find Approx. Min-Ratio Cycles

Talk Outline

Part 1: History
and ults

Part 2: Using Tree Embeddings to Find Approx. Min-Ratio Cycles

Algorithm Outline

(Input): Graph G = (V, E) with cap. u_e for each edge e, initial flow $f^{(0)}$

1. For t = 1, 2, …, $m^{1+o(1)}$ iterations:
2. Data structure maintains a spanning tree T on G.
3. Update gradients/lengths $g^{(t)}, L^{(t)} \in \mathbb{R}^E$.
4. Change tree T according to new gradients/lengths $g^{(t)}, L^{(t)}$.
5. Find cycle c represented on T via $m^{o(1)}$ off-tree edges/paths which $m^{o(1)}$-approximately minimizes $\langle g^{(t)}, c\rangle / \left\| L^{(t)} c \right\|_1$.
6. $f^{(t)} = f^{(t-1)} + c$.

Algorithm Outline

(Input): Graph $G = (V, E)$ with cap. u_e for each edge e, initial flow $f^{(0)}$

1. For $t = 1, 2, \ldots, m^{1+o(1)}$ iterations:
2. Data structure maintains a spanning tree T on G.
3. Update gradients/lengths $g^{(t)}, L^{(t)} \in \mathbb{R}^E$.
4. Change tree T according to new gradients/lengths $g^{(t)}, L^{(t)}$.
5. Find cycle c represented on T via $m^{o(1)}$ off-tree edges/paths which $m^{o(1)}$-approximately minimizes $\langle g^{(t)}, c\rangle / \|L^{(t)}c\|_1$.
6. $f^{(t)} = f^{(t-1)} + c$.

Question 1: How to decide what $g^{(t)}, L^{(t)}$ are? How to decide when to update them?

Question 2: How to maintain the tree T and find the cycle c as $g^{(t)}, L^{(t)}$ change?

Potential Reduction Interior Point Method

- Add an undirected edge e*, implicitly directed from (t,s).
- Let F = the maxflow between (s, t)
- Potential function [Kar84]: $\min_{\text{circulation } f} \Phi(f)$, where:
  - $\Phi(f) = 20m \log(F - f_{e*}) - \sum_{\text{edges } e}(\log(u_e - f_e) + \log f_e)$.

Potential Reduction Interior Point Method

- Add an undirected edge e*, implicitly directed from (t,s).
- Let F = the maxflow between (s, t)
- Potential function [Kar84]: $\min_{\text{circulation } f} \Phi(f)$, where:
  - $\Phi(f) = 20m \log(F - f_{e*}) - \sum_{\text{edges } e}(\log(u_e - f_e) + \log f_e)$.
- Trades off routing more flow (closer to F), and not getting close to capacity constraints.

Interior Point Method Details

- $\Phi(f) = 20m\ \log(F\ -\ f_{e*}) - \sum_{\text{edges } e}(\log(u_e - f_e) + \ \log f_e)$.

- Goal: reduce $\Phi(f)$ by $m^{-o(1)}$ per iteration.
- If $\Phi(f)\ \leq -O(m \log m)$ then $F\ -\ f_{e*}\ \leq m^{-O(1)}$.

Interior Point Method Details

- $\Phi(f) = 20m\ \log(F\ -\ f_{e*}) - \sum_{\text{edges } e}(\log(u_e - f_e) + \ \log f_e)$.
- Goal: reduce $\Phi(f)$ by $m^{-o(1)}$ per iteration.
- If $\Phi(f)\ \leq -O(m \log m)$ then $F\ -\ f_{e*}\ \leq m^{-O(1)}$.
- Gradient $g\ = \nabla\Phi(f)$, and lengths $L_e = \ \frac{1}{u_e - f_e} + \ \frac{1}{f_e}$.
- Update $f\ \leftarrow f +\ \eta c$ for c approx. minimizing $\langle g, c\rangle / ||Lc||_1$, scaling η

Interior Point Method Details

- $\Phi(f) = 20m\ \log(F\ -\ f_{e*}) - \sum_{\text{edges } e}(\log(u_e - f_e) + \ \log f_e)$.
- Goal: reduce $\Phi(f)$ by $m^{-o(1)}$ per iteration.
- If $\Phi(f)\ \leq -O(m \log m)$ then $F\ -\ f_{e*}\ \leq m^{-O(1)}$.
- Gradient $g\ = \nabla\Phi(f)$, and lengths $L_e =\ \frac{1}{u_e - f_e} + \frac{1}{f_e}$.
- Update $f\ \leftarrow f +\ \eta c$ for c approx. minimizing $\langle g, c\rangle / ||Lc||_1$, scaling η
- Reduces potential by $m^{-o(1)}$ per iteration, so $m^{1+o(1)}$ total iters.
- If f^* is the maxflow, choosing $c = f^* - f$ is a good direction.

Approx. MRC via Random Tree Embeddings

- Warmup: return c approx. minimizing $\langle g, c \rangle / ||Lc||_1$ in Õ(m) time.

Approx. MRC via Random Tree Embeddings

- Warmup: return c approx. minimizing $\langle g, c \rangle / ||Lc||_1$ in Õ(m) time.
- **Algorithm:** sample "random" tree T.
  - $c_T(e)$: fundamental cycle of edge e. Choose c as the best $c_T(e)$.
  - Repeat O(log n) times and take the best.

Approx. MRC via Random Tree Embeddings

- Warmup: return c approx. minimizing $\langle g, c\rangle / ||Lc||_1$ in Õ(m) time.
- **Algorithm:** sample "random" tree T.
  - $c_T(e)$: fundamental cycle of edge e. Choose c as the best $c_T(e)$.
  - Repeat O(log n) times and take the best.
- ``Random" tree: $\mathbb{E}_T[\mathrm{len}(c_T(e))] \leq \tilde{O}(1) L_e$, i.e. fundamental cycle is on average only $\tilde{O}(1)$ times longer [AKPW95,EEST08]

Approx. MRC via Random Tree Embeddings

- Warmup: return c approx. minimizing $\langle g, c\rangle / ||Lc||_1$ in Õ(m) time.
- **Algorithm:** sample "random" tree T.
  - $c_T(e)$: fundamental cycle of edge e. Choose c as the best $c_T(e)$.
  - Repeat O(log n) times and take the best.
- ``Random" tree: $\mathbb{E}_T[\mathrm{len}(c_T(e))] \leq \tilde{O}(1) L_e$, i.e. fundamental cycle is on average only $\tilde{O}(1)$ times longer [AKPW95,EEST08]
- Let c* be the optimal minimizer of $\langle g, c\rangle / ||Lc||_1$.
- Then $\mathbb{E}_T\left[\sum_{\mathrm{edges}\ e} |c_e^*| \,\mathrm{len}(c_T(e))\ \right] \leq \tilde{O}(1) ||Lc^*||_1$

Diagram for Random Tree Embedding

- $\sum_{\text{edges } e} |c_e^*| \operatorname{len}(c_T(e)) \ \leq \tilde{\mathrm{O}}(1)||Lc^*||_1$ whp. for some tree T, because we sample O(log n) trees.
- **Claim**: some cycle $c_T(e)$ is an $\tilde{\mathrm{O}}(1)$-approx. solution

Diagram for Random Tree Embedding

- $\sum_{\text{edges } e} |c_e^*| \operatorname{len}(c_T(e)) \ \leq \tilde{\mathrm{O}}(1)||Lc^*||_1$ whp. for some tree T, because we sample O(log n) trees.
- **Claim**: some cycle $c_T(e)$ is an $\tilde{\mathrm{O}}(1)$-approx. solution
- **Proof:** total gradient over all $c_T(e)$ is $\langle g, c \rangle$**,** total length is $\tilde{\mathrm{O}}(1)\left||Lc|\right|_1$

Diagram for Random Tree Embedding

- $\sum_{\text{edges } e} |c_e^*| \operatorname{len}(c_T(e)) \leq \tilde{O}(1)||Lc^*||_1$ whp. for some tree T, because we sample O(log n) trees.
- **Claim**: some cycle $c_T(e)$ is an $\tilde{O}(1)$-approx. solution
- **Proof:** total gradient over all $c_T(e)$ is $\langle g, c\rangle$, total length is $\tilde{O}(1)\left|\left|Lc\right|\right|_1$

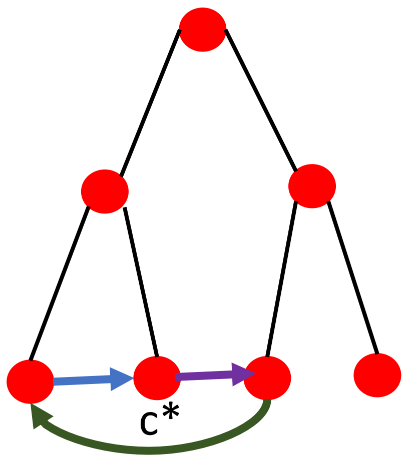


Diagram for Random Tree Embedding

- $\sum_{\text{edges } e} |c_e^*| \operatorname{len}(c_T(e)) \;\leq \tilde{O}(1)||Lc^*||_1$ whp. for some tree T, because we sample O(log n) trees.
- **Claim**: some cycle $c_T(e)$ is an $\tilde{O}(1)$-approx. solution
- **Proof:** total gradient over all $c_T(e)$ is $\langle g, c \rangle$, total length is $\tilde{O}(1)\left||Lc|\right|_1$