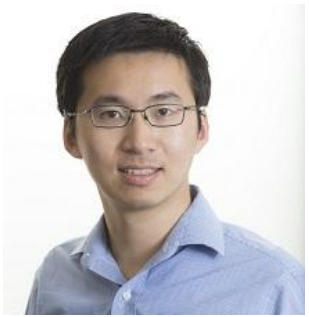
Fully Dynamic Electrical Flows: Sparse Maxflow Faster than Goldberg-Rao

Yu Gao, Yang P. Liu, Richard Peng

Georgia Tech

Stanford University

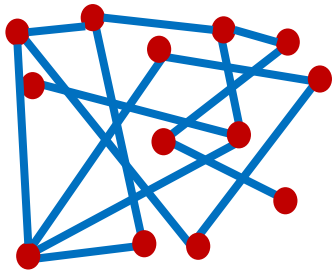


Georgia Tech &
University of Waterloo

Talk Organization

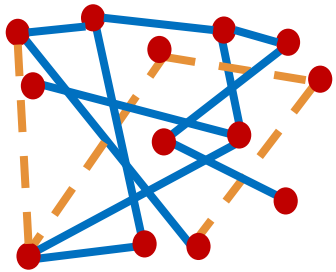
Flow Problems

Part 1



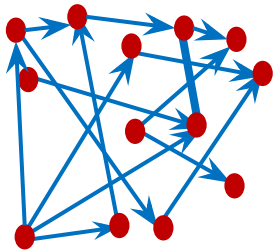
Fully Dynamic Electric Flows

Part 2



Putting it Together: Full Algorithm

Part 3



The Maximum Flow Problem

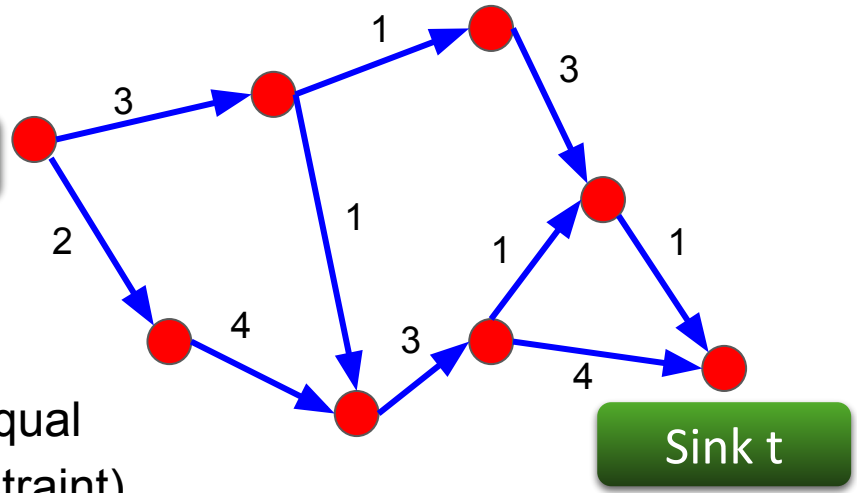
Directed graph G = (V, E) with n vertices and m edges.

Edge e has capacity u_e in [1, 2, …, U].

Source vertex s and sink t.

Route as much flow from s -> t such that flow on edge e in between [0, u_e].

Also, every vertex except s, t must have equal incoming and outgoing flow (demand constraint)



The Maximum Flow Problem

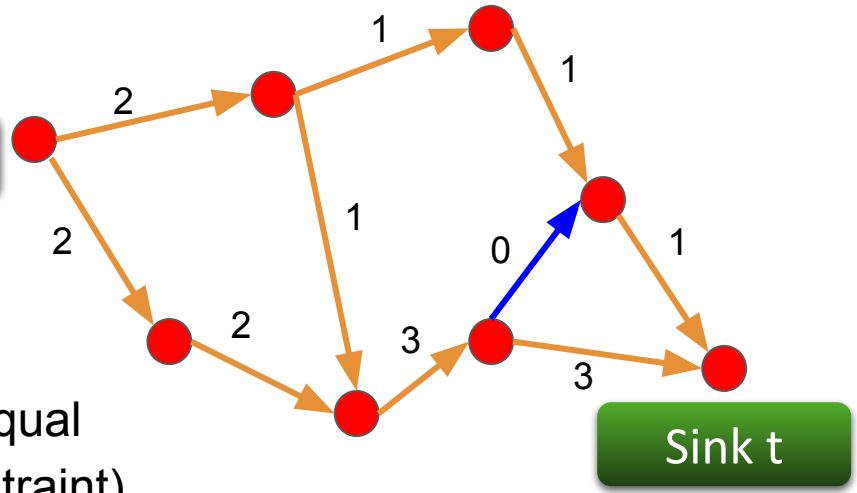
Directed graph G = (V, E) with n vertices and m edges.

Edge e has capacity u_e in [1, 2, …, U].

Source vertex s and sink t.

Route as much flow from s -> t such that flow on edge e in between [0, u_e].

Also, every vertex except s, t must have equal incoming and outgoing flow (demand constraint)



The Maximum Flow Problem

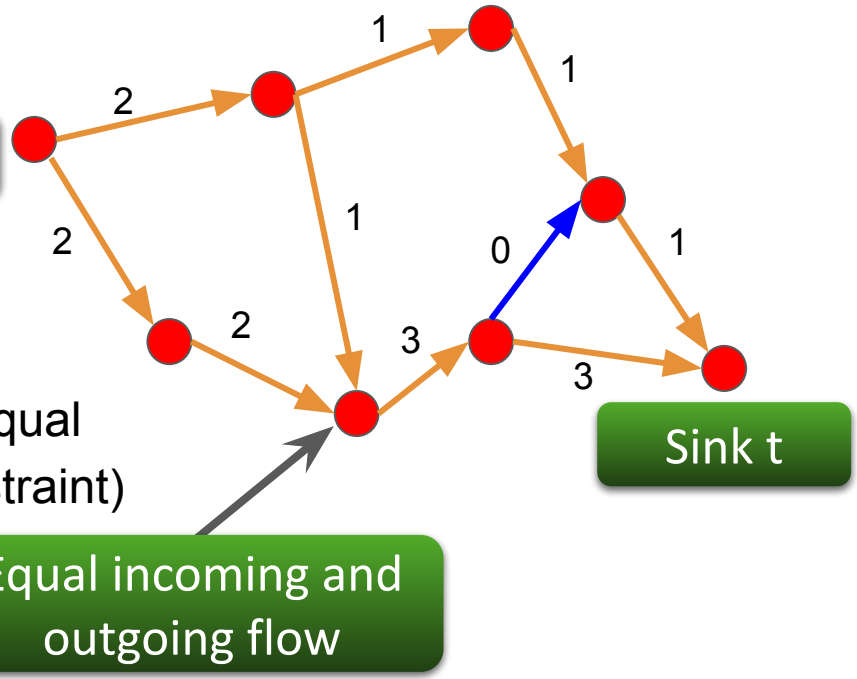
Directed graph G = (V, E) with n vertices and m edges.

Edge e has capacity u_e in [1, 2, …, U].

Source vertex s and sink t.

Route as much flow from s -> t such that flow on edge e in between [0, u_e].

Also, every vertex except s, t must have equal incoming and outgoing flow (demand constraint)



The Maximum Flow Problem

Let f in R^E denote the *flow vector*, i.e. f_e = flow on edge e.

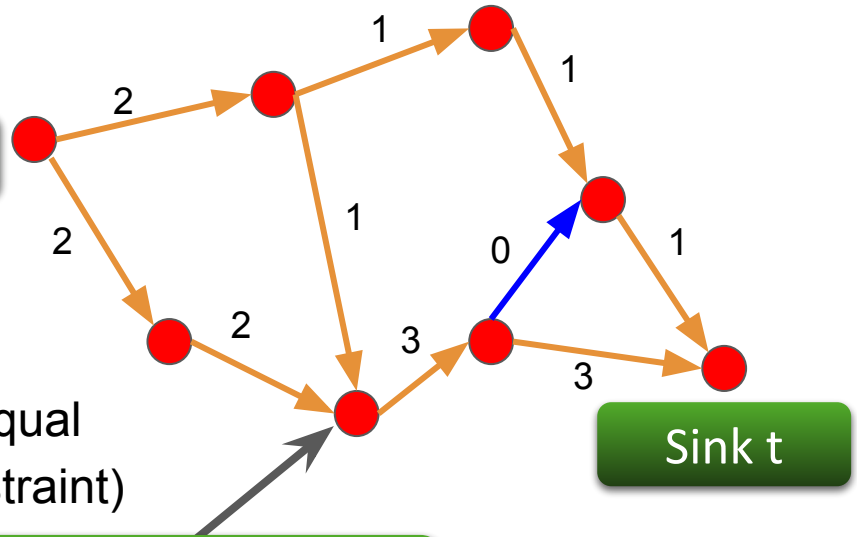
Directed graph G = (V, E) with n vertices and m edges.

Edge e has capacity u_e in [1, 2, …, U].

Source vertex s and sink t.

Route as much flow from s -> t such that flow on edge e in between [0, u_e].

Also, every vertex except s, t must have equal incoming and outgoing flow (demand constraint)



Other Variations

Decision version: instead, decide whether F units can be routed from s -> t without violating capacity constraints. **Assume this version throughout.**

Other Variations

Decision version: instead, decide whether F units can be routed from s -> t without violating capacity constraints. **Assume this version throughout.**

General demands: can we route a flow such that vertex v has net flow d_v without violating capacity constraints? **Reducible to s-t maxflow.**

We call the vector d_v for v in V the *demand vector*. Sum of d_v over v to = 0.

Other Variations

Decision version: instead, decide whether F units can be routed from s -> t without violating capacity constraints. **Assume this version throughout.**

General demands: can we route a flow such that vertex v has net flow d_v without violating capacity constraints? **Reducible to s-t maxflow.**

We call the vector d_v for v in V the *demand vector*. Sum of d_v over v to = 0.

Assume G is undirected throughout.

Why Maximum Flow?

Fundamental problem

Well studied with decades of extensive research

Historically, improvements have yielded general tools

Why Maximum Flow?

Fundamental problem

Well studied with decades of extensive research

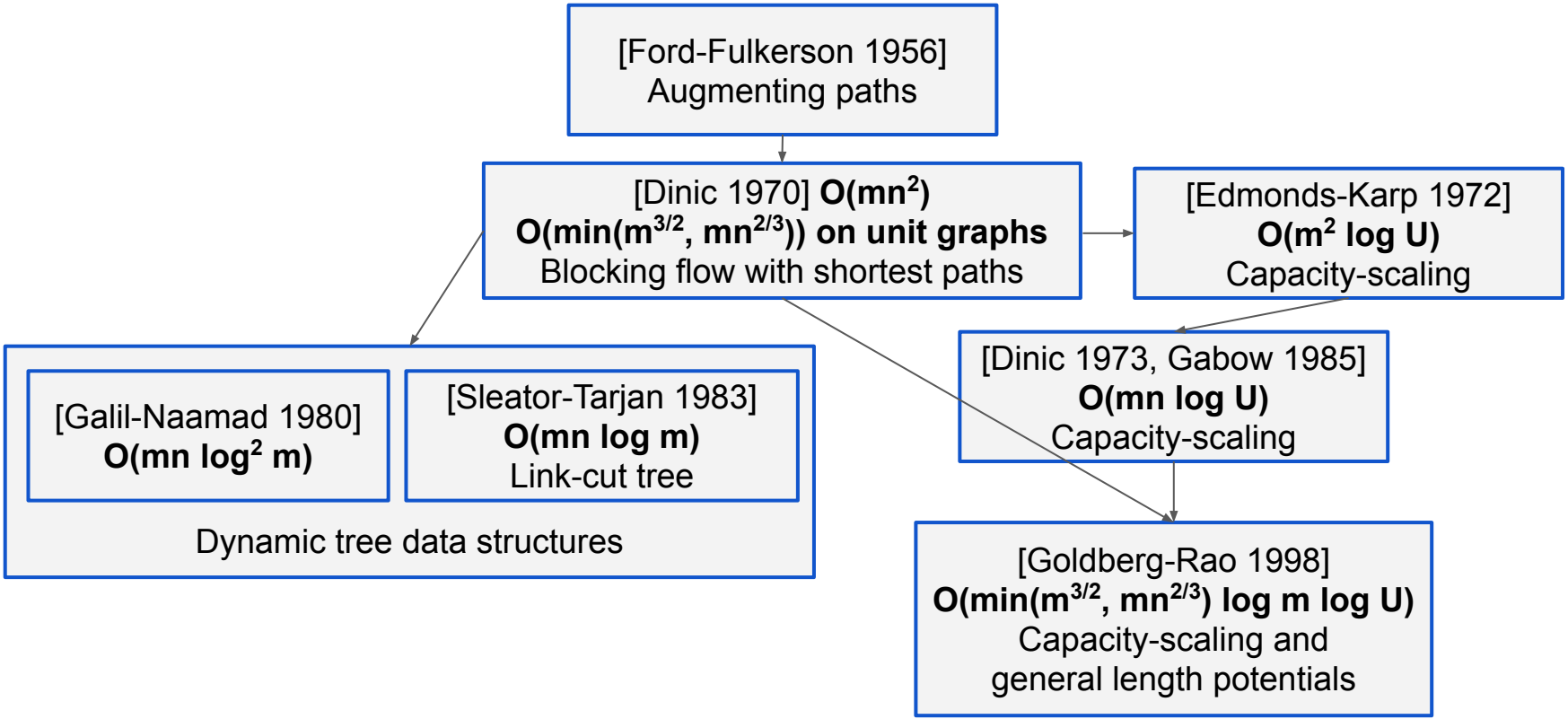
Historically, improvements have yielded general tools

Applications

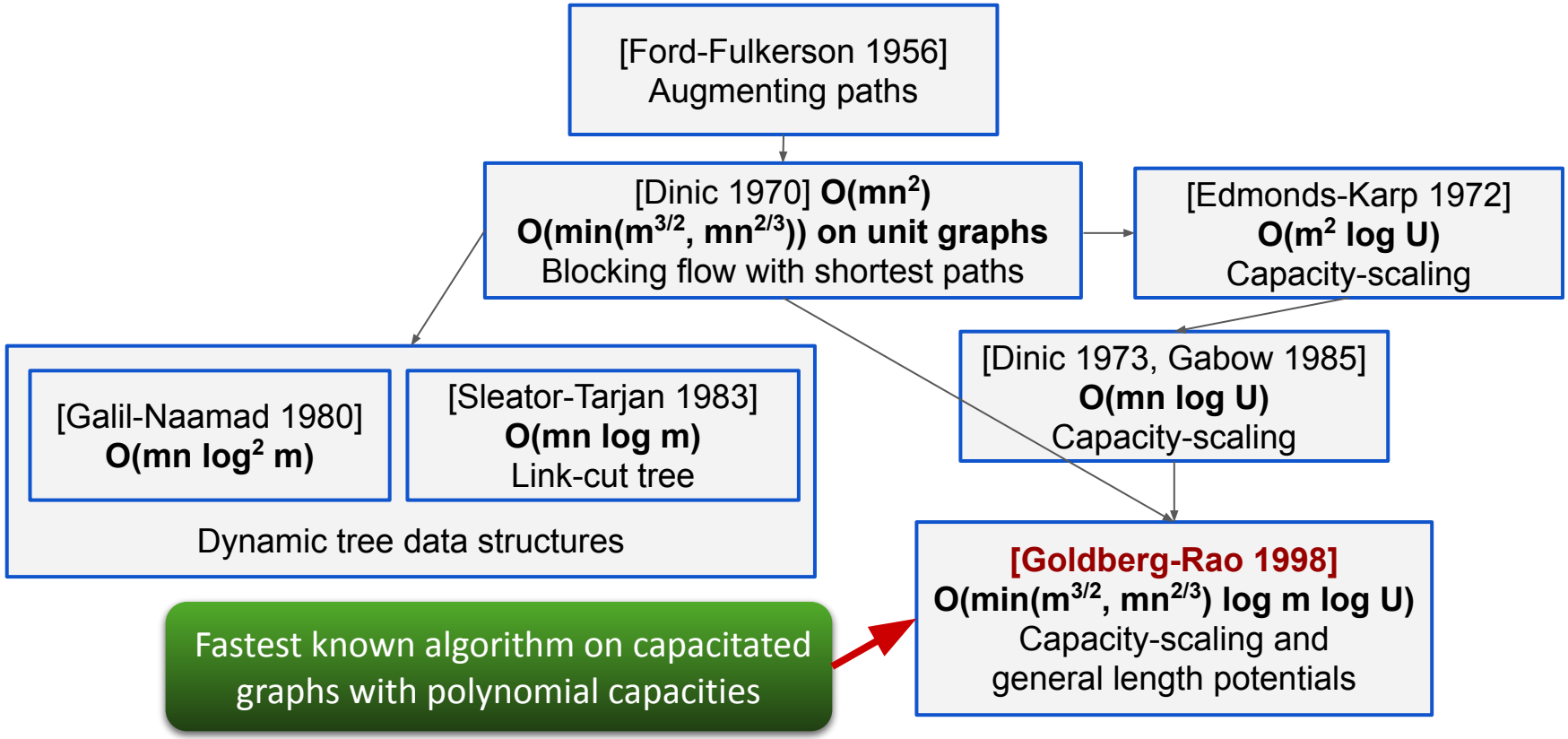
Minimum s-t cut, bipartite matching, scheduling, transportation, clustering

Methods often solve more general cases: mincost flow, negative weight shortest paths, mincost matching, optimal transport

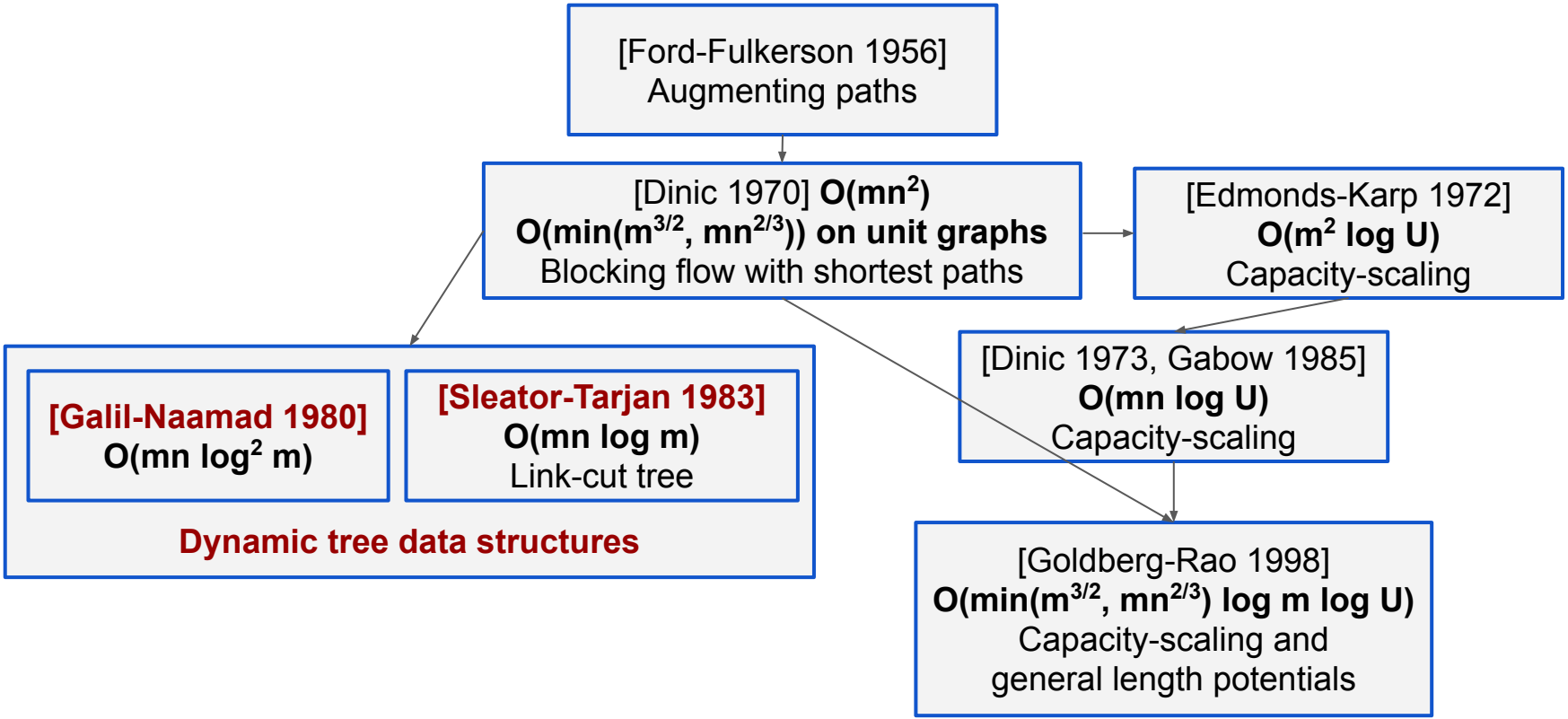
Previous Work on Maximum Flows: Classical Results



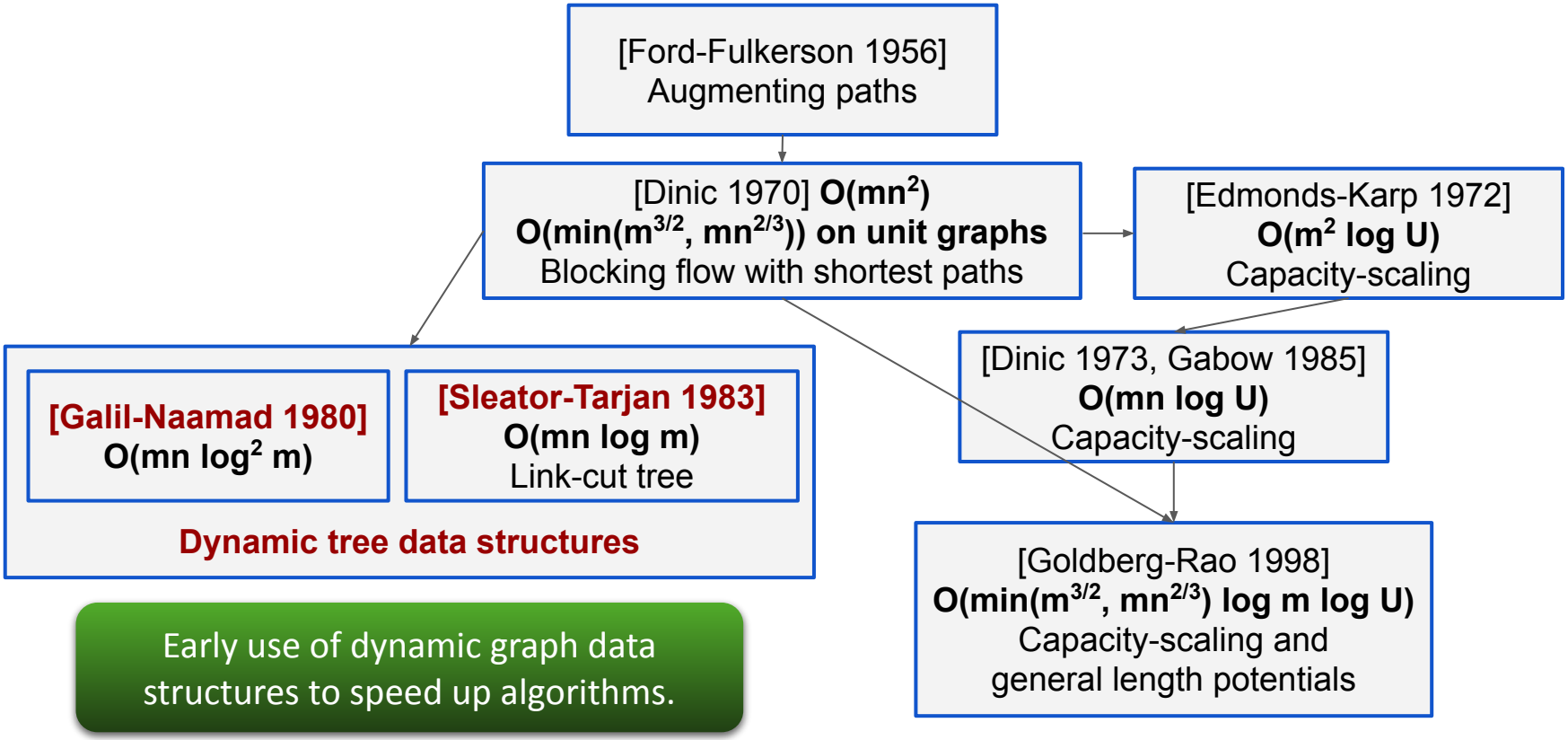
Previous Work on Maximum Flows: Classical Results



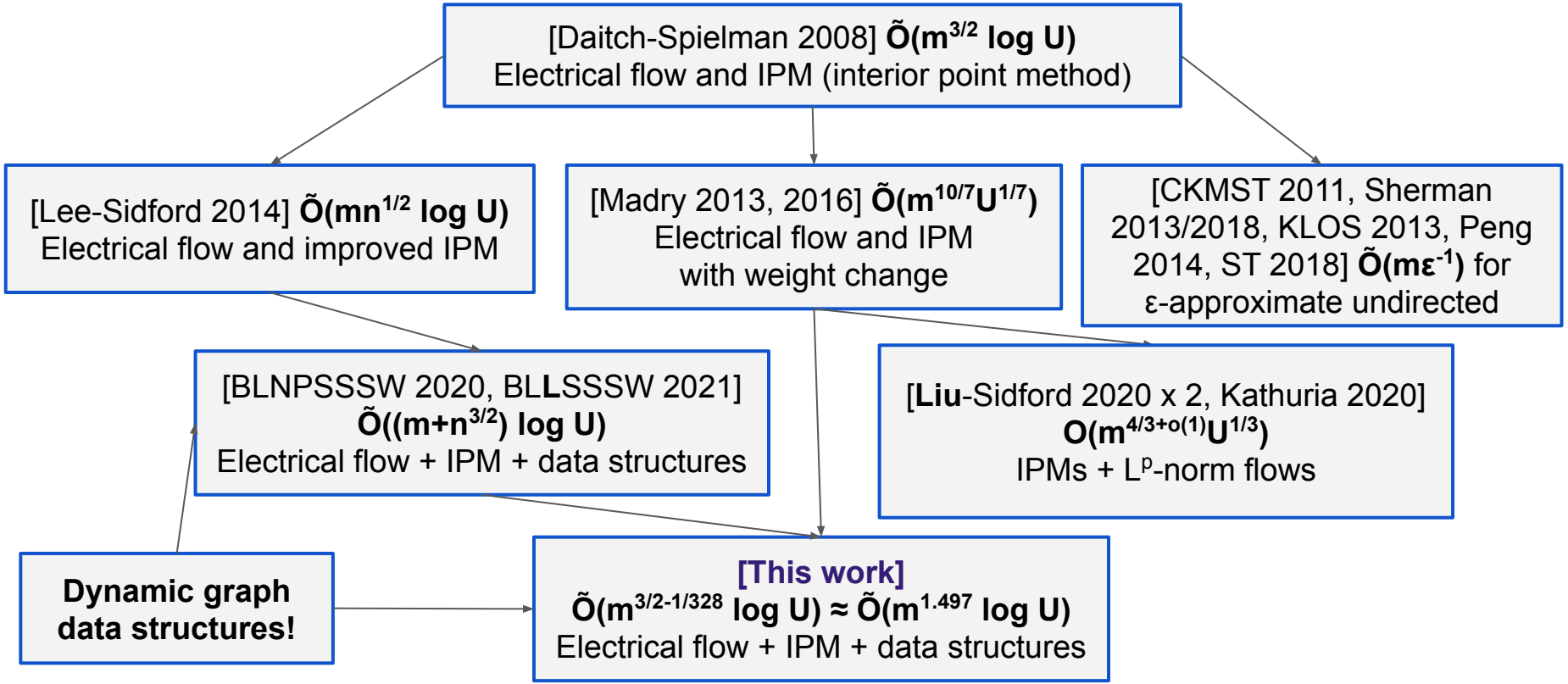
Previous Work on Maximum Flows: Classical Results



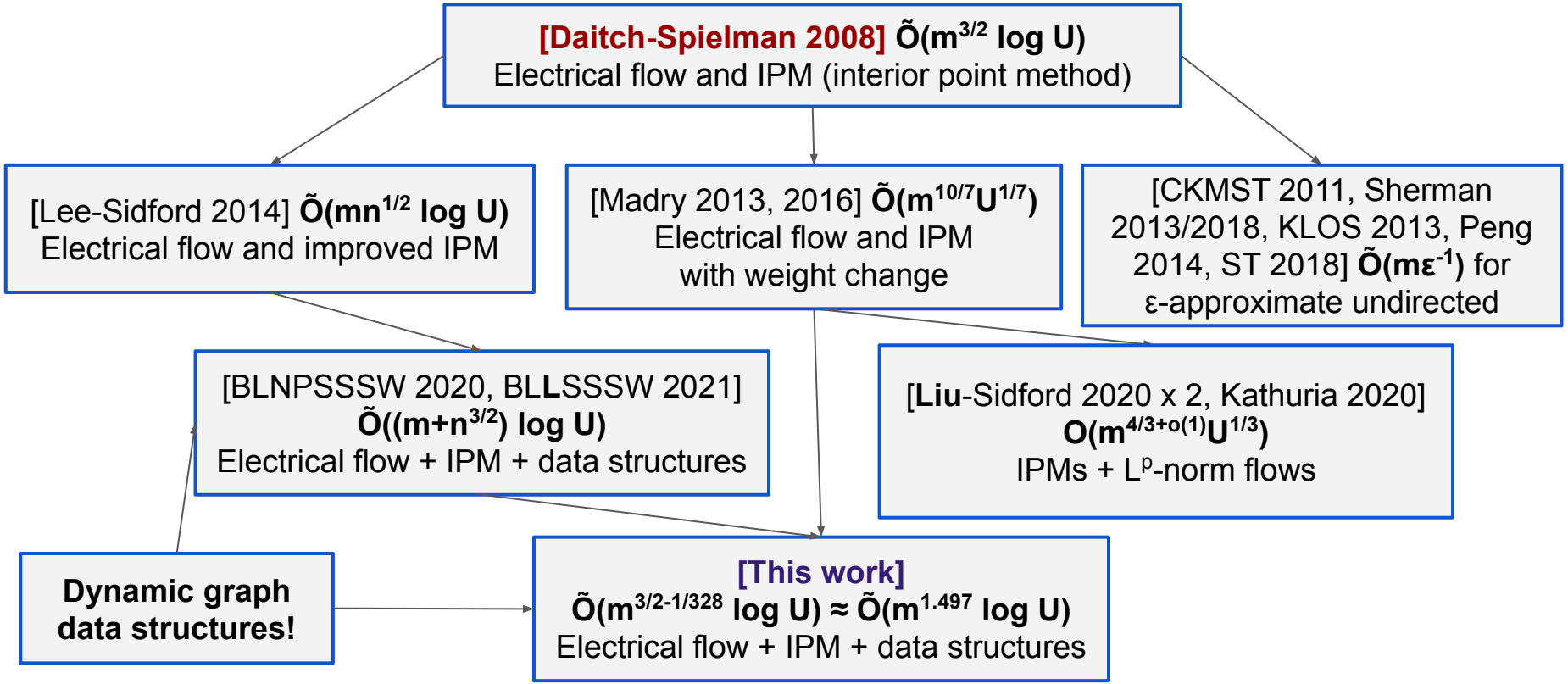
Previous Work on Maximum Flows: Classical Results



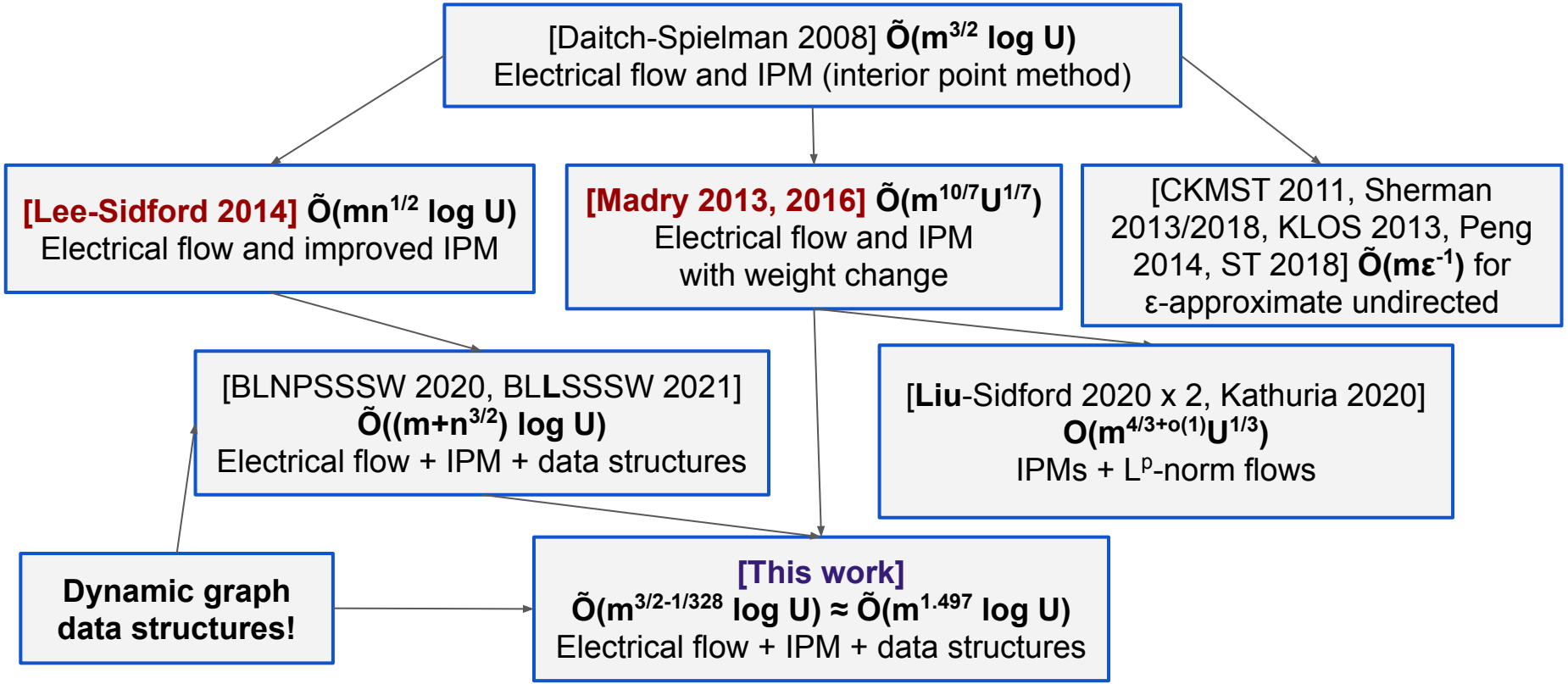
Previous Work on Maximum Flows: More Recent Work



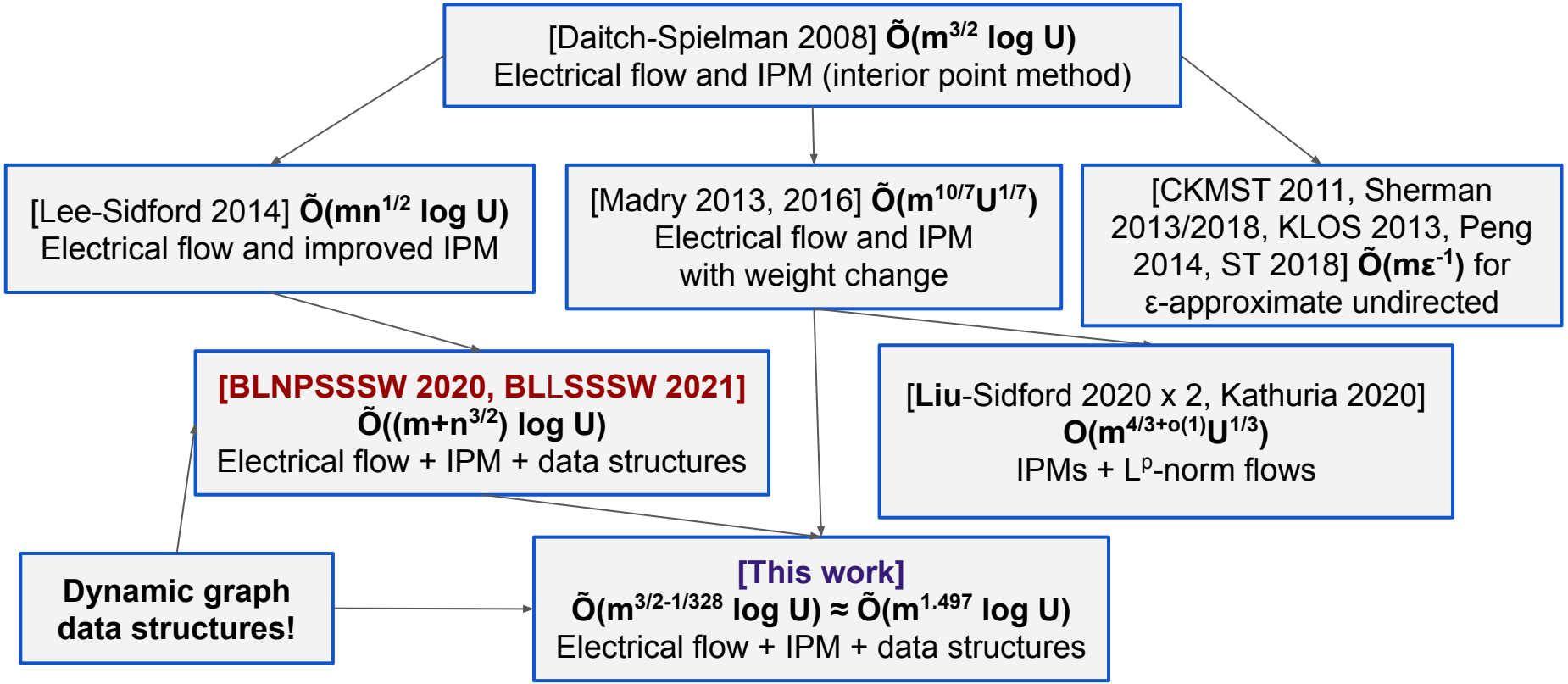
Previous Work on Maximum Flows: More Recent Work



Previous Work on Maximum Flows: More Recent Work



Previous Work on Maximum Flows: More Recent Work



Electrical Flows

Undirected graph G = (V, E), edge e has *resistance* r_e.

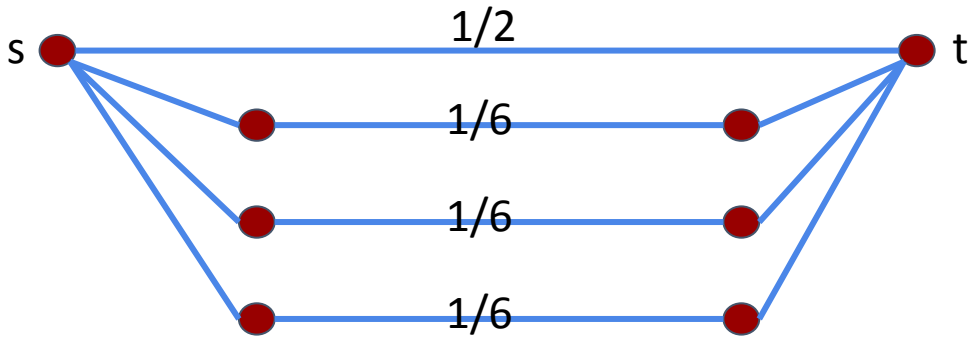
s-t electric flow is the minimum energy flow, i.e. it minimizes $\sum_e r_e f_e^2$ over all flows sending one unit from s to t. $\sum_e r_e f_e^2$ is known as the *energy.*

Electrical Flows

Undirected graph G = (V, E), edge e has *resistance* r_e.

s-t electric flow is the minimum energy flow, i.e. it minimizes $\sum_e r_e f_e^2$ over all flows sending one unit from s to t. $\sum_e r_e f_e^2$ is known as the *energy.*

Physically, corresponds to an electrical circuit where edge e is a resistor with resistance r_e, and a battery is hooked to s, t to induce voltage drop.



Why Electrical Flows?

Applications

Corresponds to solving *Laplacian systems*.

Estimate hitting probabilities for random walks, spectral graph clustering, PageRank, traffic modelling

Why Electrical Flows?

Applications

Corresponds to solving *Laplacian systems*.

Estimate hitting probabilities for random walks, spectral graph clustering, PageRank, traffic modelling

Powerful primitive

Solvable in $O(m \log^{O(1)} m)$ time [Spielman-Teng 2004]

Used in several recent advances in solving the maximum flow problem

Why Electrical Flows?

Applications

Corresponds to solving *Laplacian systems*.

Estimate hitting probabilities for random walks, spectral graph clustering, PageRank, traffic modelling

Powerful primitive

Solvable in $O(m \log \log^{O(1)} m)$ time [Jambalupati-Sidford 2020]

Used in several recent advances in solving the maximum flow problem

Dynamic Electrical Flows for Maxflow

Interior point methods (IPMs) solve linear programs in m variables by solving $\tilde{O}(m^{1/2})$ linear systems.

For maximum flow the linear systems correspond to electrical flows.

Dynamic Electrical Flows for Maxflow

Interior point methods (IPMs) solve linear programs in m variables by solving $\tilde{O}(m^{1/2})$ linear systems.

For maximum flow the linear systems correspond to electrical flows.

Concrete view [Madry 2013, 2016]: Compute maximum s-t flow by *augmenting* by $\tilde{O}(m^{1/2})$ electrical flows.

Contrasts with classical approach of augmenting *paths*.

Dynamic Electrical Flows for Maxflow

Interior point methods (IPMs) solve linear programs in m variables by solving $\tilde{O}(m^{1/2})$ linear systems.

For maximum flow the linear systems correspond to electrical flows.

Concrete view [Madry 2013, 2016]: Compute maximum s-t flow by *augmenting* by $\tilde{O}(m^{1/2})$ electrical flows.

Contrasts with classical approach of augmenting *paths*.

Basic approach: Build maximum flow from $\tilde{O}(m^{1/2})$ electrical flows.

Question: Less than O(m) per iteration?

Basic approach: add $\tilde{O}(m^{1/2})$ electrical flows to get our maximum flow.

Idea 1. Change method to reduce number of iterations: $m^{1/3+o(1)}$ for unit capacity graphs [Madry 2013/2016, **Liu**-Sidford 2020 x 2, Kathuria 2020].

Question: Less than O(m) per iteration?

Basic approach: add $\tilde{O}(m^{1/2})$ electrical flows to get our maximum flow.

Idea 1. Change method to reduce number of iterations: $m^{1/3+o(1)}$ for unit capacity graphs [Madry 2013/2016, **Liu**-Sidford 2020 x 2, Kathuria 2020].

Idea 2. Data structure for “approximate” electric flow?

The resistance of an edge is ~ inverse of the residual capacity squared.

Electric flows with 1.1-multiplicative approximate resistances suffice

Question: Less than O(m) per iteration?

Basic approach: add $\tilde{O}(m^{1/2})$ electrical flows to get our maximum flow.

Idea 1. Change method to reduce number of iterations: $m^{1/3+o(1)}$ for unit capacity graphs [Madry 2013/2016, **Liu**-Sidford 2020 x 2, Kathuria 2020].

Idea 2. Data structure for “approximate” electric flow?

The resistance of an edge is ~ inverse of the residual capacity squared.

Electric flows with 1.1-multiplicative approximate resistances suffice

Refined approach [Karmarkar, Vaidya]**:** Only update the edge resistance when it changes by > 1.1 factor -- can show this happens $\tilde{O}(m)$ total times.

Dynamic Electrical Flows for Maxflow

Refined approach [Karmarkar, Vaidya]**:** Only update the edge resistance when it changes by > 1.1 factor -- can show this happens $\tilde{O}(m)$ total times.

Dynamic Electrical Flows for Maxflow

Refined approach [Karmarkar, Vaidya]**:** Only update the edge resistance when it changes by > 1.1 factor -- can show this happens $\tilde{O}(m)$ total times.

Build a data structure that:

Initializes an undirected graph G with resistances.

Supports resistance updates.

Can report all edges in e with large flow/energy in an s-t electric flow.

Dynamic Electrical Flows for Maxflow

Refined approach [Karmarkar, Vaidya]**:** Only update the edge resistance when it changes by > 1.1 factor -- can show this happens $\tilde{O}(m)$ total times.

Build a data structure that:

Initializes an undirected graph G with resistances.

Supports resistance updates.

Can report all edges in e with large flow/energy in an s-t electric flow.

At a high level, such a data structure with sublinear m^{1-c} amortized runtime improves over $O(m^{3/2} \log U)$ time maxflow.

Our Results: Heavy Hitters for Dynamic Electric s-t Flows

Randomized data structure against oblivious adversaries.

Guarantee: The energy of the s-t electric flow is <= 1 at all times.

Operations:

- Initialize(G = (V, E), r in R^E, ε) -- Graph G, resistance r_e on edge e, accuracy ε
- Update(e, r^{new}) -- changes the resistances of edge e to r^{new}.
- Locate() -- Returns $O(\varepsilon^{-2})$ edges containing all edges e with *energy* $r_e f_e^2 >= \varepsilon^2$. (Here f_e is *any* (1+ε)-approximate s-t electric flow)

Our Results: Heavy Hitters for Dynamic Electric s-t Flows

Randomized data structure against oblivious adversaries.

Guarantee: The energy of the s-t electric flow is <= 1 at all times.

Operations:

- Initialize(G = (V, E), r in R^E, ε) -- Graph G, resistance r_e on edge e, accuracy ε
- Update(e, r^{new}) -- changes the resistances of edge e to r^{new}.
- Locate() -- Returns $O(\varepsilon^{-2})$ edges containing all edges e with *energy* $r_e f_e^2 >= \varepsilon^2$. (Here f_e is *any* (1+ε)-approximate s-t electric flow)

Theorem **[GLP21]**: There is a data structure supporting these operations in armotized sublinear $m^{0.99}\varepsilon^{-O(1)}$ time per query.

Our Results: Maxflow Faster Than $m^{1.5}$ log U

Theorem **[GLP21]**: Maximum flow on graphs with capacities in [1, U] can be solved in time $\tilde{O}(m^{3/2-1/328} \log U)$.

Our Results: Maxflow Faster Than $m^{1.5}$ log U

Theorem **[GLP21]**: Maximum flow on graphs with capacities in [1, U] can be solved in time $\tilde{O}(m^{3/2-1/328} \log U)$.

First improvement to the $O(m^{1.5} \log m \log U)$ time algorithm of Goldberg-Rao 1998 on sparse graphs.

Our Results: Maxflow Faster Than $m^{1.5}$ log U

Theorem **[GLP21]**: Maximum flow on graphs with capacities in [1, U] can be solved in time $\tilde{O}(m^{3/2-1/328} \log U)$.

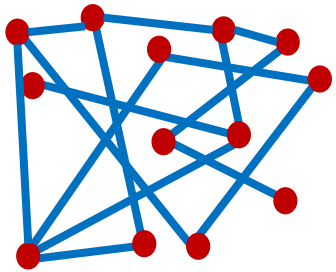
First improvement to the $O(m^{1.5} \log m \log U)$ time algorithm of Goldberg-Rao 1998 on sparse graphs.

Approach: use the previous data structure to implicitly add $\tilde{O}(m^{1/2})$ electrical flows to get the final maxflow, implementing each iteration in sublinear time.

Talk Organization

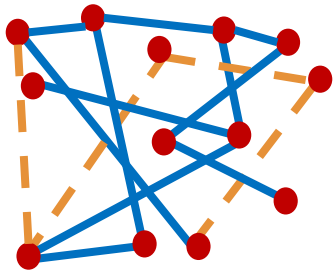
Flow Problems

Part 1



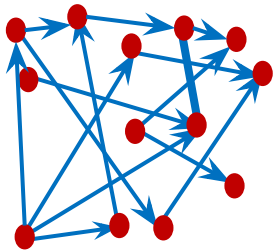
Fully Dynamic Electric Flows

Part 2



Putting it Together: Full Algorithm

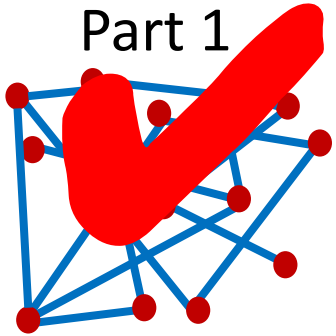
Part 3



Talk Organization

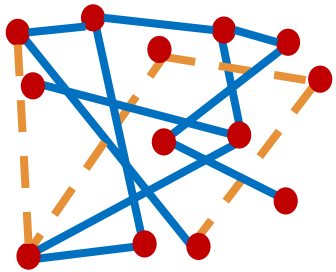
Flow Problems

Part 1



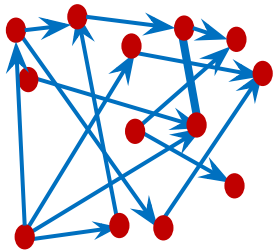
Fully Dynamic Electric Flows

Part 2



Putting it Together: Full Algorithm

Part 3



Dynamic Graph Data Structures and Vertex Sparsifiers

Goal: Build data structures that achieve efficient (eg. sublinear) update time for processing graphs that are changing.

Examples: edge connectivity, flows, shortest paths and reachability in directed graphs, effective resistances

Dynamic Graph Data Structures and Vertex Sparsifiers

Goal: Build data structures that achieve efficient (eg. sublinear) update time for processing graphs that are changing.

Examples: edge connectivity, flows, shortest paths and reachability in directed graphs, effective resistances

Approaches: Graph decompositions -- reduce the problem on a large graph to a similar problem on a smaller graph

Referred to as *vertex sparsification*: reduce the number of vertices while maintaining important quantities, i.e. edge connectivity or effective resistance.

Previous Work on Dynamic Graph Data Structures

Edge connectivity: under a graph with edge insertions and deletions, decide whether vertices s-t are c-edge-connected, i.e. there are c disjoint paths.

[Fre85, GI91b, GI91a, WT92, EGIN97, Fre97, HK97, HT97, HK99, Tho00, HdLT01, KKM13, Wul13, KRKPT16, NS17, NSW17, Wul17, HRT18, CGL+19].

Recently, c-edge-connectivity vertex sparsifiers: [CDKL**L**PSV20, JS20] and $n^{o(1)}$ deterministic worst-case update time for constant c.

Previous Work on Dynamic Graph Data Structures

Edge connectivity: under a graph with edge insertions and deletions, decide whether vertices s-t are c-edge-connected, i.e. there are c disjoint paths.

[Fre85, GI91b, GI91a, WT92, EGIN97, Fre97, HK97, HT97, HK99, Tho00, HdLT01, KKM13, Wul13, KRKPT16, NS17, NSW17, Wul17, HRT18, CGL+19].

Recently, c-edge-connectivity vertex sparsifiers: [CDKL**L**PSV20, JS20] and $n^{o(1)}$ deterministic worst-case update time for constant c.

Flow approximations and shortest paths: [CGHPS20, GRST20] sublinear time $n^{o(1)}$ approximate shortest paths and maxflow.

Previous Work on Dynamic Graph Data Structures

Edge connectivity: under a graph with edge insertions and deletions, decide whether vertices s-t are c-edge-connected, i.e. there are c disjoint paths.

[Fre85, GI91b, GI91a, WT92, EGIN97, Fre97, HK97, HT97, HK99, Tho00, HdLT01, KKM13, Wul13, KRKPT16, NS17, NSW17, Wul17, HRT18, CGL+19].

Recently, c-edge-connectivity vertex sparsifiers: [CDKL**L**PSV20, JS20] and $n^{o(1)}$ deterministic worst-case update time for constant c.

Flow approximations and shortest paths: [CGHPS20, GRST20] sublinear time $n^{o(1)}$ approximate shortest paths and maxflow.

Dynamic effective resistances: [DGGP19, CGHPS20] sublinear time (1+ε)-approximate pairwise effective resistance in dynamic graph.

Effective Resistances and Schur Complements

Given a graph G = (V, E) be B denote the (m x n) edge-vertex incidence matrix.

Laplacian: For R = diag(r) (diagonal matrix of resistances), $L = B^T R^{-1} B$.

Effective Resistances and Schur Complements

Given a graph G = (V, E) be B denote the (m x n) edge-vertex incidence matrix.

Laplacian: For R = diag(r) (diagonal matrix of resistances), $L = B^T R^{-1} B$.

Effective resistance between s-t: $\chi_{st} L^{-1} \chi_{st}$.

Effective Resistances and Schur Complements

Given a graph G = (V, E) be B denote the (m x n) edge-vertex incidence matrix.

Laplacian: For R = diag(r) (diagonal matrix of resistances), $L = B^T R^{-1} B$.

Effective resistance between s-t: $\chi_{st} L^{-1} \chi_{st}$.

χ_{st} = demand routing 1 unit from s -> t. +1 is s-coordinate, -1 in t-coordinate.

Effective Resistances and Schur Complements

Effective resistance between s-t: $\chi_{st} L^{-1} \chi_{st} = \chi_{st} (B^T R^{-1} B)^{-1} \chi_{st}$.

Effective Resistances and Schur Complements

Effective resistance between s-t: $\chi_{st} L^{-1} \chi_{st} = \chi_{st} (B^T R^{-1} B)^{-1} \chi_{st}$.

Intuition: this is the spectral form induced by the Laplacian inverse.

Vertex sparsifier onto C containing {s, t}: construct a Laplacian on C whose inverse approximates the inverse of L on vectors supported on C.

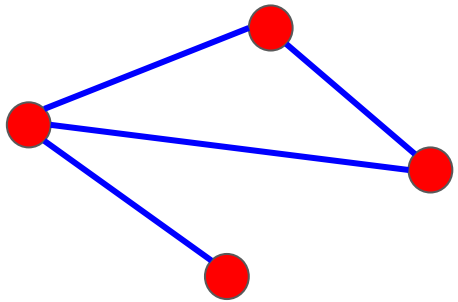
Effective Resistances and Schur Complements

Effective resistance between s-t: $\chi_{st} L^{-1} \chi_{st} = \chi_{st} (B^T R^{-1} B)^{-1} \chi_{st}$.

Intuition: this is the spectral form induced by the Laplacian inverse.

Vertex sparsifier onto C containing {s, t}: construct a Laplacian on C whose inverse approximates the inverse of L on vectors supported on C.

Concretely: matrix L_C such that for all vectors d on C, $d^T L^{-1} d \approx d^T L_C^{-1} d$.



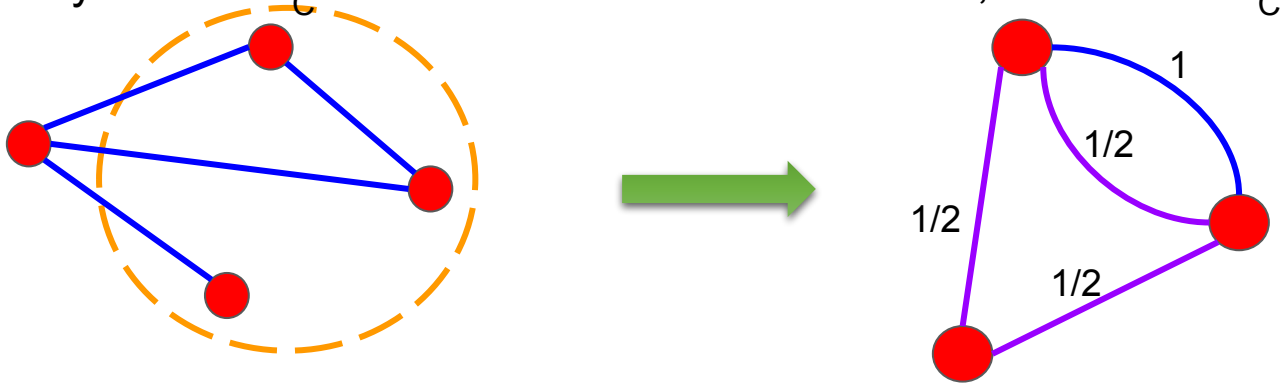
Effective Resistances and Schur Complements

Effective resistance between s-t: $\chi_{st}L^{-1}\chi_{st} = \chi_{st}(B^TR^{-1}B)^{-1}\chi_{st}$.

Intuition: this is the spectral form induced by the Laplacian inverse.

Vertex sparsifier onto C containing {s, t}: construct a Laplacian on C whose inverse approximates the inverse of L on vectors supported on C.

Concretely: matrix L_C such that for all vectors d on C, $d^TL^{-1}d \approx d^TL_C^{-1}d$.



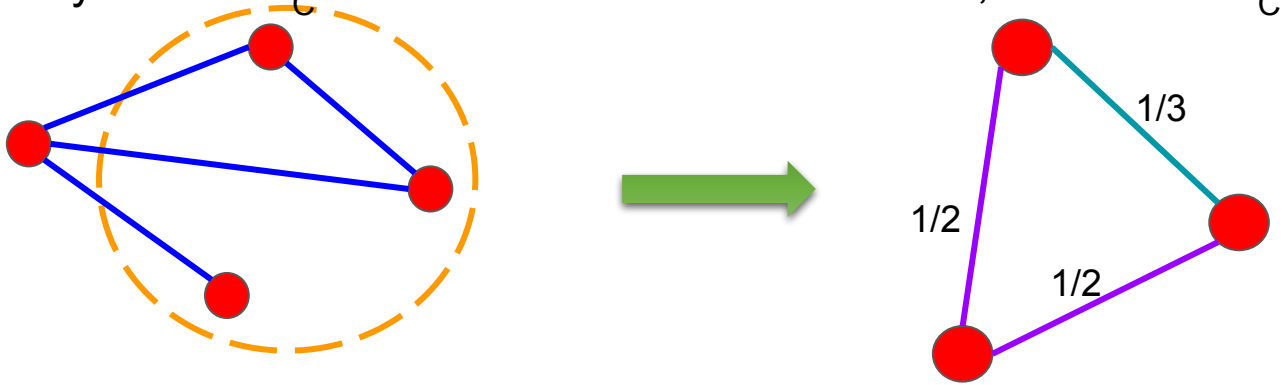
Effective Resistances and Schur Complements

Effective resistance between s-t: $\chi_{st}L^{-1}\chi_{st} = \chi_{st}(B^TR^{-1}B)^{-1}\chi_{st}$.

Intuition: this is the spectral form induced by the Laplacian inverse.

Vertex sparsifier onto C containing {s, t}: construct a Laplacian on C whose inverse approximates the inverse of L on vectors supported on C.

Concretely: matrix L_C such that for all vectors d on C, $d^TL^{-1}d \approx d^TL_C^{-1}d$.



Effective Resistances and Schur Complements

Concretely: matrix L_C such that for all vectors d on C, $d^T L^{-1} d \approx d^T L_C^{-1} d$.

Effective Resistances and Schur Complements

Concretely: matrix L_C such that for all vectors d on C, $d^T L^{-1} d \approx d^T L_C^{-1} d$.

Matrix L_C exists that preserves the inverse form *exactly*: Schur complement.

$$L = \begin{bmatrix} L_{FF} & L_{FC} \\ L_{CF} & L_{CC} \end{bmatrix} \quad \text{and} \quad SC(L, C) := L_{CC} - L_{CF} L_{FF}^{-1} L_{FC}$$

Effective Resistances and Schur Complements

Concretely: matrix L_C such that for all vectors d on C, $d^T L^{-1} d \approx d^T L_C^{-1} d$.

Matrix L_C exists that preserves the inverse form *exactly*: Schur complement.

$$L = \begin{bmatrix} L_{FF} & L_{FC} \\ L_{CF} & L_{CC} \end{bmatrix} \quad \text{and} \quad SC(L, C) := L_{CC} - L_{CF} L_{FF}^{-1} L_{FC}$$

Essentially Gaussian elimination on blocks.

Effective Resistances and Schur Complements

Concretely: matrix L_C such that for all vectors d on C, $d^T L^{-1} d \approx d^T L_C^{-1} d$.

Matrix L_C exists that preserves the inverse form *exactly*: Schur complement.

$$L = \begin{bmatrix} L_{FF} & L_{FC} \\ L_{CF} & L_{CC} \end{bmatrix} \quad \text{and} \quad SC(L, C) := L_{CC} - L_{CF} L_{FF}^{-1} L_{FC}$$

Can show that SC(L, C) is a Laplacian matrix.

Cholesky factorization: $$L^{-1} = \begin{bmatrix} I & -L_{FF}^{-1} L_{FC} \\ 0 & I \end{bmatrix} \begin{bmatrix} L_{FF}^{-1} & 0 \\ 0 & SC(L, C)^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ L_{CF} L_{FF}^{-1} & I \end{bmatrix}$$

Dynamic Schur Complements by Random Walks

Approach: solve dynamic effective resistance by maintaining approximate dynamic SC (need approximate b/c the true SC is dense)

Dynamic Schur Complements by Random Walks

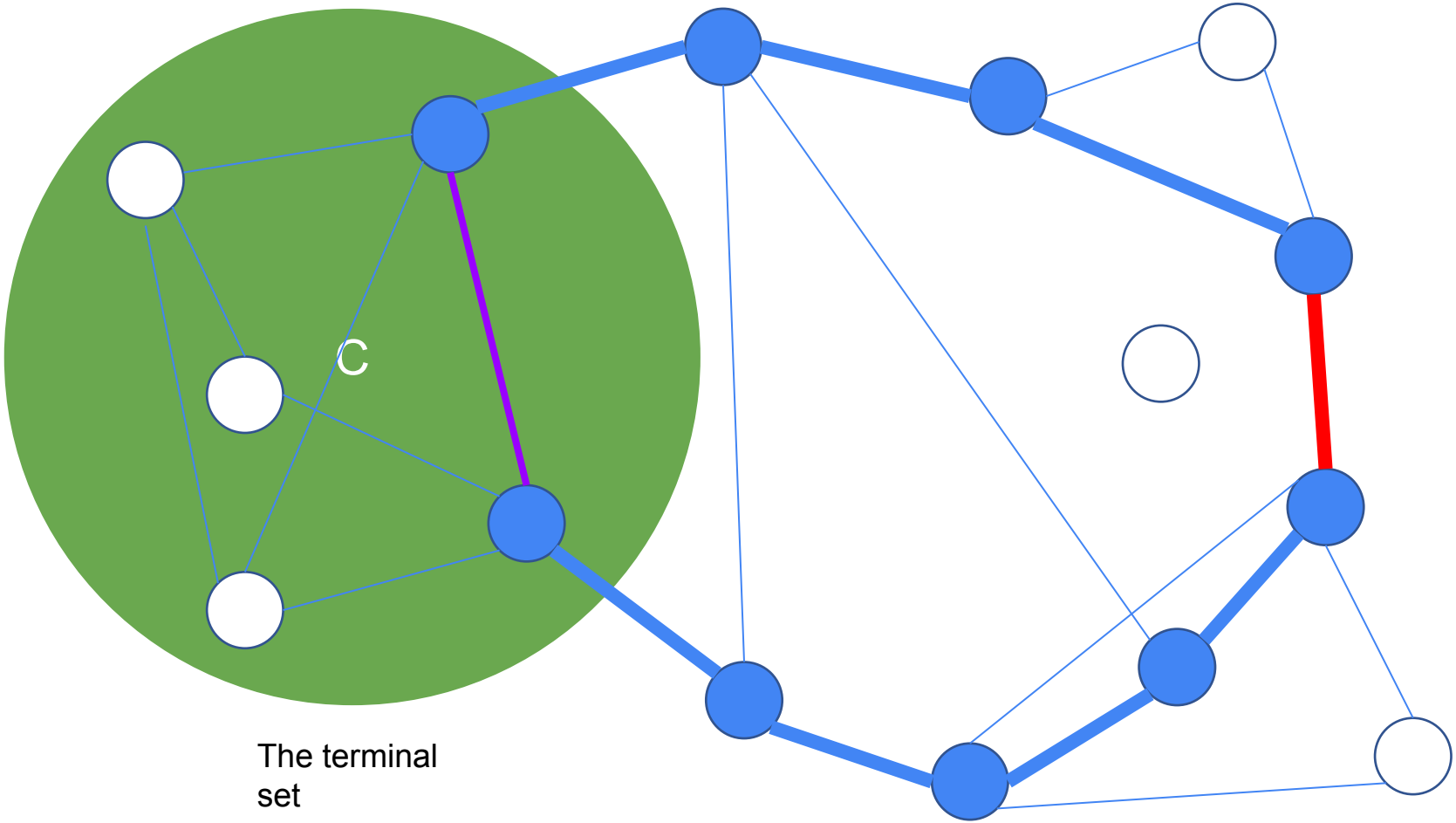
Approach: solve dynamic effective resistance by maintaining approximate dynamic SC (need approximate b/c the true SC is dense)

[DGGP19]: Maintain SC approximation by sampling random walks.

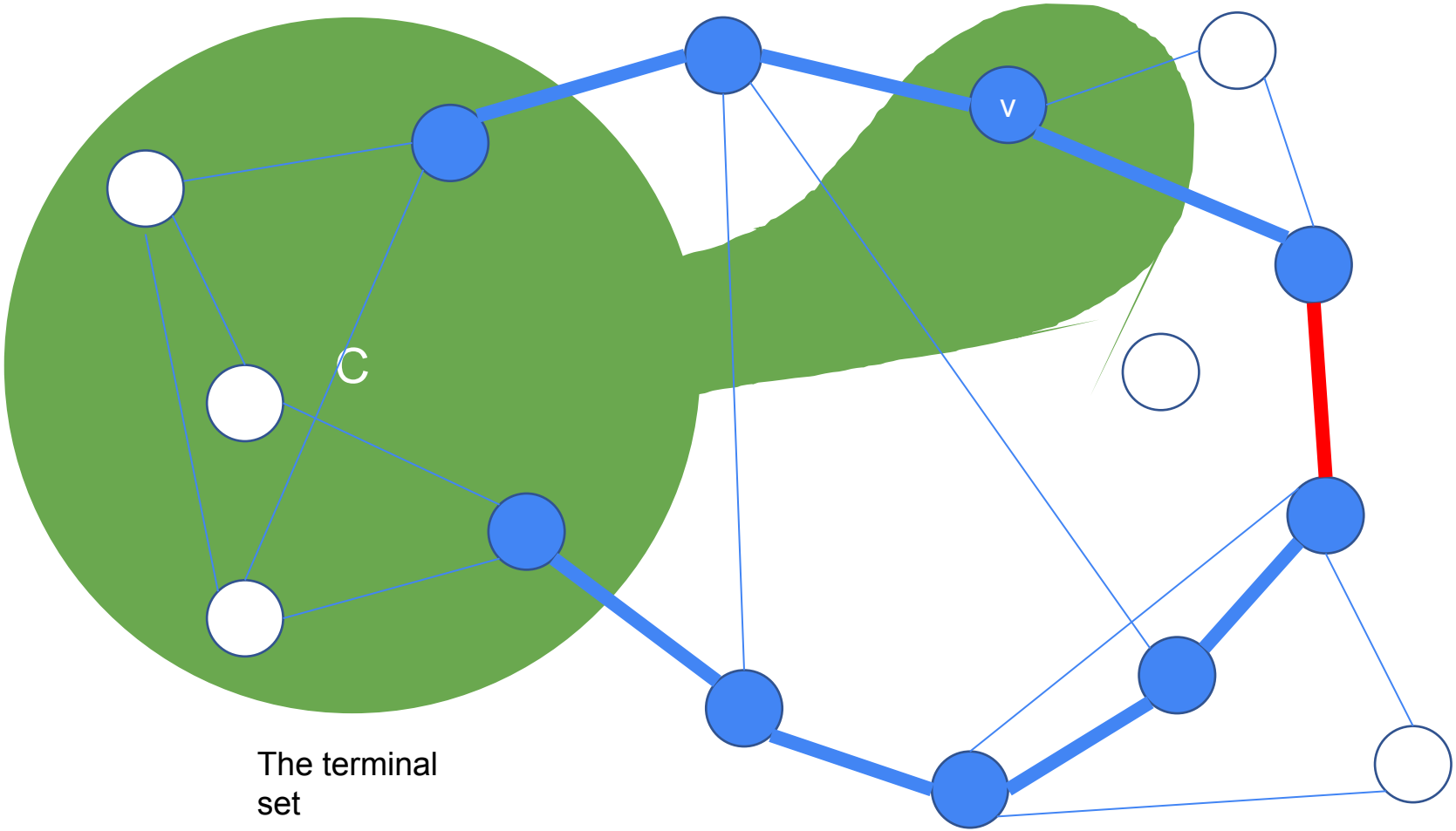
Theorem: Let C be a subset of vertices of a graph G = (V, E, r). Consider the following procedure. For each edge e = (u, v) in E, repeat p = $O(\varepsilon^{-2} \log n)$ times:

1. Run a random walk from u (resp. v) with exit probability proportional to r^{-1} until it hits C, say at t_1 and t_2.
2. Let W be the set of edges on the walk when connected using edge (u, v).
3. Add edge (t_1, t_2) to a graph H with resistance = $p \sum_{f \in W} r_f$

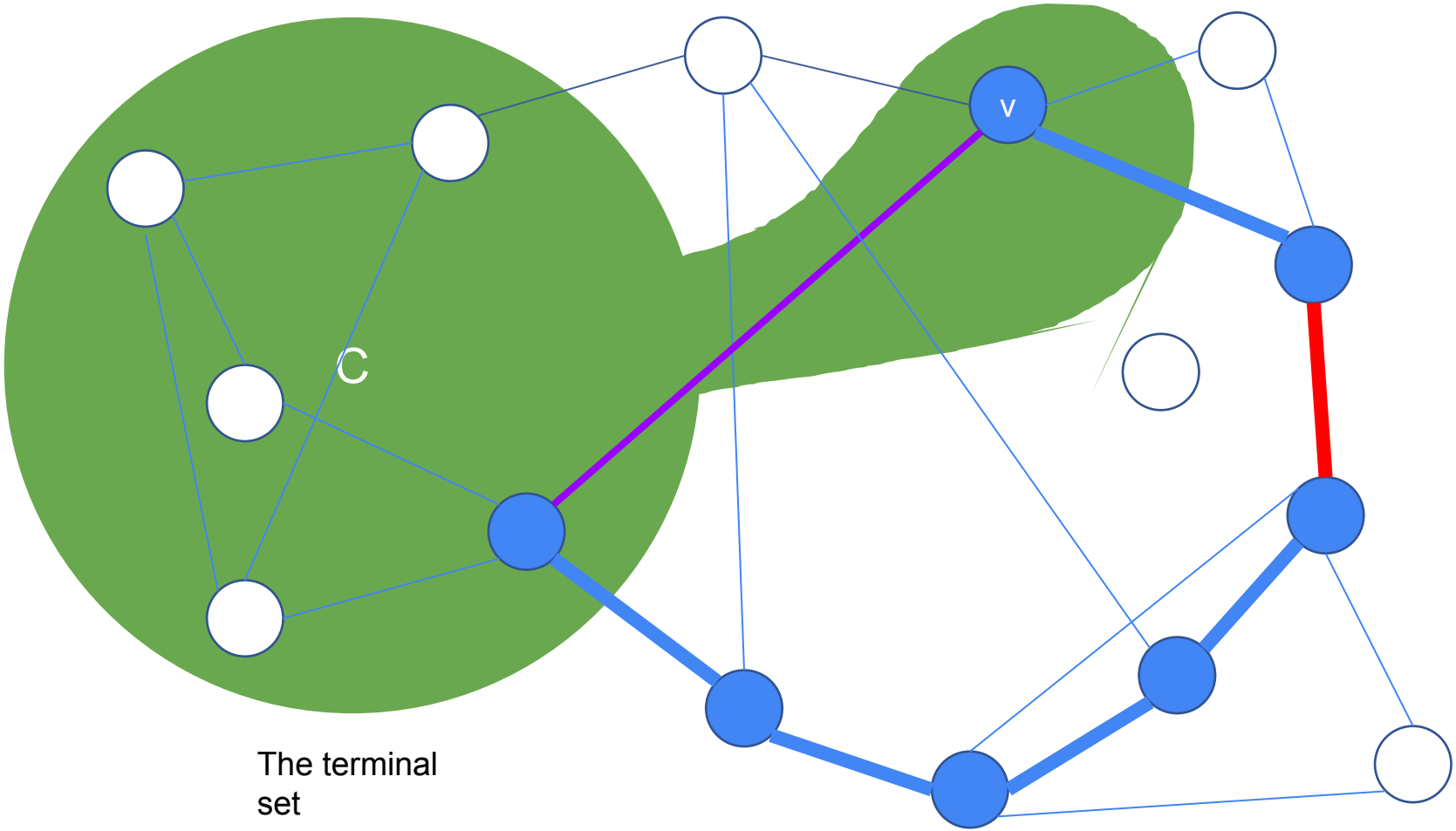
Then H is a (1+ε)-approximation spectrally of SC(L(G), C).



The terminal set



The terminal set



The terminal set

Dynamic Schur Complements by Random Walks

Algorithm for dynamic SC under resistance changes:

1. Initialize random walks from each edge e in G.
2. When edge e = (u, v) has resistance changed, add both endpoints u, v into the set C (the *terminals*). Shortcut random walks accordingly.
3. Update the resistance of edge e (which now fully inside C)

Dynamic Schur Complements by Random Walks

Algorithm for dynamic SC under resistance changes:

1. Initialize random walks from each edge e in G.
2. When edge e = (u, v) has resistance changed, add both endpoints u, v into the set C (the *terminals*). Shortcut random walks accordingly.
3. Update the resistance of edge e (which now fully inside C)

When effective resistance of s-t is queried:

1. Add s, t to the set C (the terminals).
2. Return $\chi_{st}H^{-1}\chi_{st}$, where H is the approximation to SC(L, C) maintained.

Step 2 of this can take time $\tilde{O}(|C|\varepsilon^{-2})$ if we use a $1+\varepsilon$ sparsifier of H.

Electric Flow Concepts

For a demand vector d, we say that a flow f *routes* d if $B^T f = d$.

Electric *potentials* for demand d is given by $\Phi = L^{-1}d = (B^T R^{-1} B)^{-1} d$.

Electric flow: $f = R^{-1}B\Phi = R^{-1}B(B^T R^{-1} B)^{-1} d$.

Ohm's Law: for e = (u, v) we have $f_e = (\Phi_u - \Phi_v)/r_e$.

Formal Dynamic Electric Flow Setup

Problem: Given initial graph G = (V, E, r), source s, sink t, accuracy ε,

1. Update: change resistance of edge e.
2. Query: for the unit s-t electric flow $f = R^{-1}B(B^TR^{-1}B)^{-1}\chi_{st}$, guaranteed that the energy is <= 1, i.e. $\|R^{1/2}f\|_2^2 = \sum_{e\in E} r_e f_e^2 \leq 1$, return a subset S of edge that contains all edges e with energy $r_e f_e^2 \geq \varepsilon^2$, and $|S| = O(\varepsilon^{-2})$.

Formal Dynamic Electric Flow Setup

Problem: Given initial graph G = (V, E, r), source s, sink t, accuracy ε,

1. Update: change resistance of edge e.
2. Query: for the unit s-t electric flow $f = R^{-1}B(B^TR^{-1}B)^{-1}\chi_{st}$, guaranteed that the energy is <= 1, i.e. $\|R^{1/2}f\|_2^2 = \sum_{e\in E} r_e f_e^2 \leq 1$, return a subset S of edge that contains all edges e with energy $r_e f_e^2 \geq \varepsilon^2$, and $|S| = O(\varepsilon^{-2})$.

Harder than dynamic ER because we have to report the high energy edges -- in dynamic ER we are told the pair of vertices in the ER query.

Reduction to Maintaining Energy of Subsets

We wish to find coordinates of the vector $R^{1/2}f$ that are >= ε, given $|R^{1/2}f|_2$ <= 1.

Reduction to Maintaining Energy of Subsets

We wish to find coordinates of the vector $R^{1/2}f$ that are >= ε, given $|R^{1/2}f|_2$ <= 1.

L2 heavy hitter [KNPW11]: there is a sketch matrix Q that is $\tilde{O}(\varepsilon^{-2})$ x n, such that for any vector y in R^n, and vector v satisfying |v - Qy|_inf <= ε/100, we can find all coordinates of y that are >= ε in time $\tilde{O}(\varepsilon^{-2})$. Q has all entries in {-1, 0, 1}.

Reduction to Maintaining Energy of Subsets

We wish to find coordinates of the vector $R^{1/2}f$ that are >= ε, given $|R^{1/2}f|_2$ <= 1.

L2 heavy hitter [KNPW11]: there is a sketch matrix Q that is $\tilde{O}(\varepsilon^{-2})$ x n, such that for any vector y in R^n, and vector v satisfying |v - Qy|_inf <= ε/100, we can find all coordinates of y that are >= ε in time $\tilde{O}(\varepsilon^{-2})$. Q has all entries in {-1, 0, 1}.

Use this for $y = R^{1/2}f$ -- need to approximately maintain
$QR^{1/2}f = QR^{-1/2}B(B^TR^{-1}B)^{-1}\chi_{st}$ under changes to R.

Reduction to Maintaining Energy of Subsets

Maintain: $QR^{1/2}f = QR^{-1/2}B(B^{T}R^{-1}B)^{-1}\chi_{st}$ under changes to R.

Reduction to Maintaining Energy of Subsets

Maintain: $QR^{1/2}f = QR^{-1/2}B(B^TR^{-1}B)^{-1}\chi_{st}$ under changes to R.

$$QR^{-1/2}B(B^TR^{-1}B)^{-1}\chi_{st} = QR^{-1/2}B\begin{bmatrix} I & -L_{FF}^{-1}L_{FC} \\ 0 & I \end{bmatrix}\begin{bmatrix} L_{FF}^{-1} & 0 \\ 0 & SC(L,C)^{-1} \end{bmatrix}\begin{bmatrix} I & 0 \\ -L_{CF}L_{FF}^{-1} & I \end{bmatrix}\chi_{st}$$

$$= QR^{-1/2}B\begin{bmatrix} -L_{FF}^{-1}L_{FC} \\ I \end{bmatrix}SC(L,C)^{-1}\chi_{st}$$

Reduction to Maintaining Energy of Subsets

Maintain: $QR^{1/2}f = QR^{-1/2}B(B^TR^{-1}B)^{-1}\chi_{st}$ under changes to R.

$$QR^{-1/2}B(B^TR^{-1}B)^{-1}\chi_{st} = QR^{-1/2}B\begin{bmatrix} I & -L_{FF}^{-1}L_{FC} \\ 0 & I \end{bmatrix}\begin{bmatrix} L_{FF}^{-1} & 0 \\ 0 & SC(L,C)^{-1} \end{bmatrix}\begin{bmatrix} I & 0 \\ -L_{CF}L_{FF}^{-1} & I \end{bmatrix}\chi_{st}$$

$$= QR^{-1/2}B\begin{bmatrix} -L_{FF}^{-1}L_{FC} \\ I \end{bmatrix}SC(L,C)^{-1}\chi_{st}$$

Demand projection

Potentials on terminal

$\tilde{O}(\varepsilon^{-2})$ length $|C|$ vectors

Length $|C|$ vector

Reduction to Maintaining Energy of Subsets

Maintain: $QR^{1/2}f$

Plan: approximately maintain both quantities dynamically.
We'll start by discussing the potentials, then the demand projection.

$$QR^{-1/2}B(B^TR^{-1}B)^{-1}\begin{bmatrix} I & 0 \\ -L_{CF}L_{FF}^{-1} & I \end{bmatrix}\chi_{st}$$

$$= QR^{-1/2}B\begin{bmatrix} -L_{FF}^{-1}L_{FC} \\ I \end{bmatrix} SC(L,C)^{-1}\chi_{st}$$

Demand projection — Potentials on terminal

$\tilde{O}(\varepsilon^{-2})$ length $|C|$ vectors

Length $|C|$ vector

Maintaining Potentials on Terminals

Maintain: $SC(L, C)^{-1}\chi_{st.}$

Maintaining Potentials on Terminals

Maintain: $SC(L, C)^{-1}\chi_{st.}$

Solution: Directly use the dynamic Schur complement from [DGGP19] for effective resistance!

Prove that (1+ε)-approximate Schur complement suffices for our purposes.

Maintaining Demand Projections

Maintain: $\begin{bmatrix} L_{CF} L_{FF}^{-1} & I \end{bmatrix} B^\top R^{-1/2} Q^\top$ <- transpose of earlier expression.

Maintaining Demand Projections

Maintain: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} B^\top R^{-1/2} Q^\top$ <- transpose of earlier expression.

Let q be a column of Q^T, there are $\tilde{O}(\varepsilon^{-2})$ of these. Recall q is in $\{-1, 0, 1\}^n$.

Let $d = B^T R^{-1/2} q$, so d is a demand vector.

Maintaining Demand Projections

Maintain: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} B^\top R^{-1/2} Q^\top$ <- transpose of earlier expression.

Let q be a column of Q^T, there are $\tilde{O}(\varepsilon^{-2})$ of these. Recall q is in $\{-1, 0, 1\}^n$.

Let $d = B^T R^{-1/2} q$, so d is a demand vector.

Maintain: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$ -- for simplicity say d fixed, and L changes (via resistance updates in the graph).

Maintaining Demand Projections

Maintain: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} B^\top R^{-1/2} Q^\top$ <- transpose of earlier expression.

Let q be a column of Q^T, there are $\tilde{O}(\varepsilon^{-2})$ of these. Recall q is in $\{-1, 0, 1\}^n$.

Let d = $B^T R^{-1/2} q$, so d is a demand vector.

Maintain: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$ -- for simplicity say d fixed, and L changes (via resistance updates in the graph).

Interpretation of above quantity via random walks -- intuitively you think of d as a mass over vertices of G, and you “random walk” that mass onto C via exit probability proportional to inverse of resistances.

Demand Projections via Random Walks

Quantity: $\begin{bmatrix} L_{CF} L_{FF}^{-1} & I \end{bmatrix} d$

Demand Projections via Random Walks

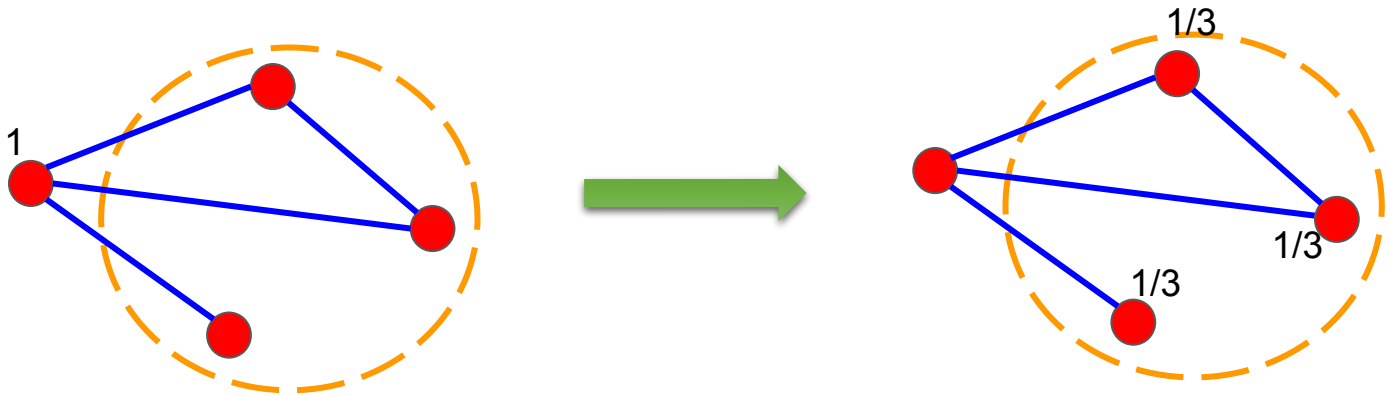
Quantity: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$

Linear in d, so let's understand what happens if $d = 1_v$.

Demand Projections via Random Walks

Quantity: $\begin{bmatrix} L_{CF} L_{FF}^{-1} & I \end{bmatrix} d$

Linear in d, so let's understand what happens if d = 1_v.



Reductions to Ensure Stability

Quantity: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$, need an approximation of each coordinate up to +-ε.

Reductions to Ensure Stability

Quantity: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$, need an approximation of each coordinate up to +-ε.

Recall $d = B^T R^{-1/2} q$, where q in {-1, 0, 1}^n.

If R has some small entries, then d can have very large entries, and affect the approximation / stability.

Reductions to Ensure Stability

Quantity: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$, need an approximation of each coordinate up to +-ε.

Recall $d = B^T R^{-1/2} q$, where q in {-1, 0, 1}^n.

If R has some small entries, then d can have very large entries, and affect the approximation / stability.

Lemma: If $r_e <= \varepsilon^2$, then energy of e, i.e. $r_e f_e^2 <= \varepsilon^2$.

Reductions to Ensure Stability

Quantity: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$, need an approximation of each coordinate up to +-ε.

Recall $d = B^T R^{-1/2} q$, where q in {-1, 0, 1}^n.

If R has some small entries, then d can have very large entries, and affect the approximation / stability.

Lemma: If $r_e <= \varepsilon^2$, then energy of e, i.e. $r_e f_e^2 <= \varepsilon^2$.

Proof: Because f is a unit s-t electric flow, we know $f_e <= 1$. So $r_e f_e^2 <= r_e <= \varepsilon^2$.

Reductions to Ensure Stability

Quantity: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$, need an approximation of each coordinate up to +-ε.

Recall $d = B^T R^{-1/2} q$, where q in {-1, 0, 1}^n.

If R has some small entries, then d can have very large entries, and affect the approximation / stability.

Lemma: If $r_e <= \varepsilon^2$, then energy of e, i.e. $r_e f_e^2 <= \varepsilon^2$.

Proof: Because f is a unit s-t electric flow, we know $f_e <= 1$. So $r_e f_e^2 <= r_e <= \varepsilon^2$.

So we assume that $d_v <= \deg(v)\varepsilon^{-1}$ from now on. Intuitively, think of as O(1).

Approach 1: Project Demands by Random Walks

Maintain: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$

Interpretation of above quantity via random walks -- intuitively you think of d as a mass over vertices of G, and you “random walk” that mass onto C via exit probability proportional to inverse of resistances.

Approach 1: Project Demands by Random Walks

Maintain: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$

Interpretation of above quantity via random walks -- intuitively you think of d as a mass over vertices of G, and you “random walk” that mass onto C via exit probability proportional to inverse of resistances.

Idea: Use random walks sampled from each vertex to simulate this!

Approach 1: Project Demands by Random Walks

Maintain: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$

Interpretation of above quantity via random walks -- intuitively you think of d as a mass over vertices of G, and you “random walk” that mass onto C via exit probability proportional to inverse of resistances.

Idea: Use random walks sampled from each vertex to simulate this!

Issue: Say d_v = O(1), and we sample p walks from v until they hit C. “Variance” of this sample is O(1) per sample, so 1/p when you average over s walks.

There are m vertices / edges to sample from, so the sum of variance is m/p: huge!

Estimate *Change* in Projected Demands by Sampling

Maintain: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$. From now on, we will denote as $\pi^C(d)$.

Read as “projection of d onto C”.

Estimate *Change* in Projected Demands by Sampling

Maintain: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$. From now on, we will denote as $\pi^C(d)$.

Estimate *Change* in Projected Demands by Sampling

Maintain: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$. From now on, we will denote as $\pi^C(d)$.

Idea: Instead estimate the change in $\pi^C(d)$ under terminal insertions to C, and rebuild every couple of iterations once error accumulates.

Estimate *Change* in Projected Demands by Sampling

Maintain: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$. From now on, we will denote as $\pi^C(d)$.

Idea: Instead estimate the change in $\pi^C(d)$ under terminal insertions to C, and rebuild every couple of iterations once error accumulates.

Fact: $\pi^{C\cup v}(d) - \pi^C(d) = \left[\pi^{C\cup v}(d)\right]_v \cdot (\chi_v - \pi^C(\chi_v))$.

Estimate *Change* in Projected Demands by Sampling

Maintain: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$. From now on, we will denote as $\pi^C(d)$.

Idea: Instead estimate the change in $\pi^C(d)$ under terminal insertions to C, and rebuild every couple of iterations once error accumulates.

Fact: $\pi^{C\cup v}(d) - \pi^C(d) = \left[\pi^{C\cup v}(d)\right]_v \cdot (\chi_v - \pi^C(\chi_v))$.

Coordinate v of the vector.

Indicator vector of vertex v.

Estimate *Change* in Projected Demands by Sampling

Maintain: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$. From now on, we will denote as $\pi^C(d)$.

Idea: Instead estimate the change in $\pi^C(d)$ under terminal insertions to C, and rebuild every couple of iterations once error accumulates.

Fact: $\pi^{C\cup v}(d) - \pi^C(d) = \left[\pi^{C\cup v}(d)\right]_v \cdot (\chi_v - \pi^C(\chi_v))$.

Proof: Consider first sampling a random walk from a vertex u to {C U v}. Now, take all the mass at v and continue random walking it until it hits C. This is a valid simulation of random walking from u -> C only. Equating these two formulations gives the result.

Estimate *Change* in Projected Demands by Sampling

Maintain: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$. From now on, we will denote as $\pi^C(d)$.

Fact: $\pi^{C\cup v}(d) - \pi^C(d) = \left[\pi^{C\cup v}(d)\right]_v \cdot (\chi_v - \pi^C(\chi_v))$.

Estimate *Change* in Projected Demands by Sampling

Maintain: $\left[\, L_{CF} L_{FF}^{-1} \quad I \,\right] d$. From now on, we will denote as $\pi^C(d)$.

Fact: $\pi^{C \cup v}(d) - \pi^C(d) = \left[\pi^{C \cup v}(d)\right]_v \cdot (\chi_v - \pi^C(\chi_v))$.

Maintain by sampling random walks up front and dynamically shorten.

Maintain by sampling fresh random walks from v to C.

Estimate *Change* in Projected Demands by Sampling

Maintain: $\begin{bmatrix} L_{CF}L_{FF}^{-1} & I \end{bmatrix} d$. From now on, we will denote as $\pi^C(d)$.

Fact: $\pi^{C\cup v}(d) - \pi^C(d) = \left[\pi^{C\cup v}(d)\right]_v \cdot (\chi_v - \pi^C(\chi_v))$.

Maintain by sampling random walks up front and dynamically shorten.

Maintain by sampling fresh random walks from v to C.

Intuitive analysis: Both terms have variance ~cong(v)/p, where cong(v) = # of random walks passing through v and p = oversampling. Much smaller error!

Rebuilding Terminals and Projected Demands

Let’s say sampling the change induces error δ, and desired error ε.

Rebuilding Terminals and Projected Demands

Let’s say sampling the change induces error δ, and desired error ε.

Support $\delta^{-1}\varepsilon$ terminal additions, then recompute $\left[L_{CF} L_{FF}^{-1} \quad I \right] d = \pi^{C}(d)$ exactly.

Rebuilding Terminals and Projected Demands

Let’s say sampling the change induces error δ, and desired error ε.

Support $\delta^{-1}\varepsilon$ terminal additions, then recompute $\left[L_{CF} L_{FF}^{-1} \quad I \right] d = \pi^{C}(d)$ exactly.

Doable with a single Laplacian system solve.

Amortized time $\delta m \varepsilon^{-3}$, as there are $\tilde{O}(\varepsilon^{-2})$ test vectors.

Pick $\delta << \varepsilon$, and trade off parameters properly (increase oversampling parameter).

Adaptive Adversaries and Checker Data Structure

As described, previous algorithm’s randomness might affect future queries.

Adaptive Adversaries and Checker Data Structure

As described, previous algorithm's randomness might affect future queries.

We fix this by wrapping it inside a separate algorithm *Checker* which independently estimates the flow on edges with a separate dynamic SC.

Need to carefully ensure that extra edges returned by the heavy hitter are not "accepted" by *Checker*, i.e. returned to the main IPM to update.

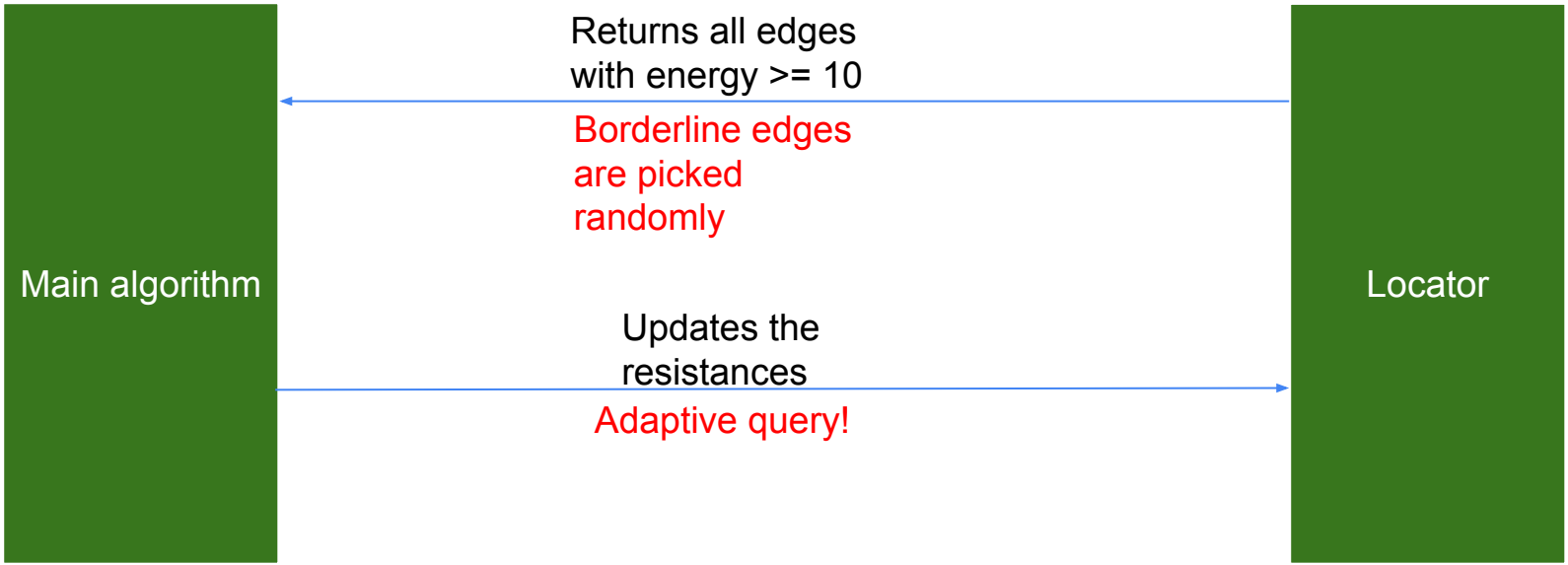
Adaptive Adversaries and Checker Data Structure

As described, previous algorithm's randomness might affect future queries.

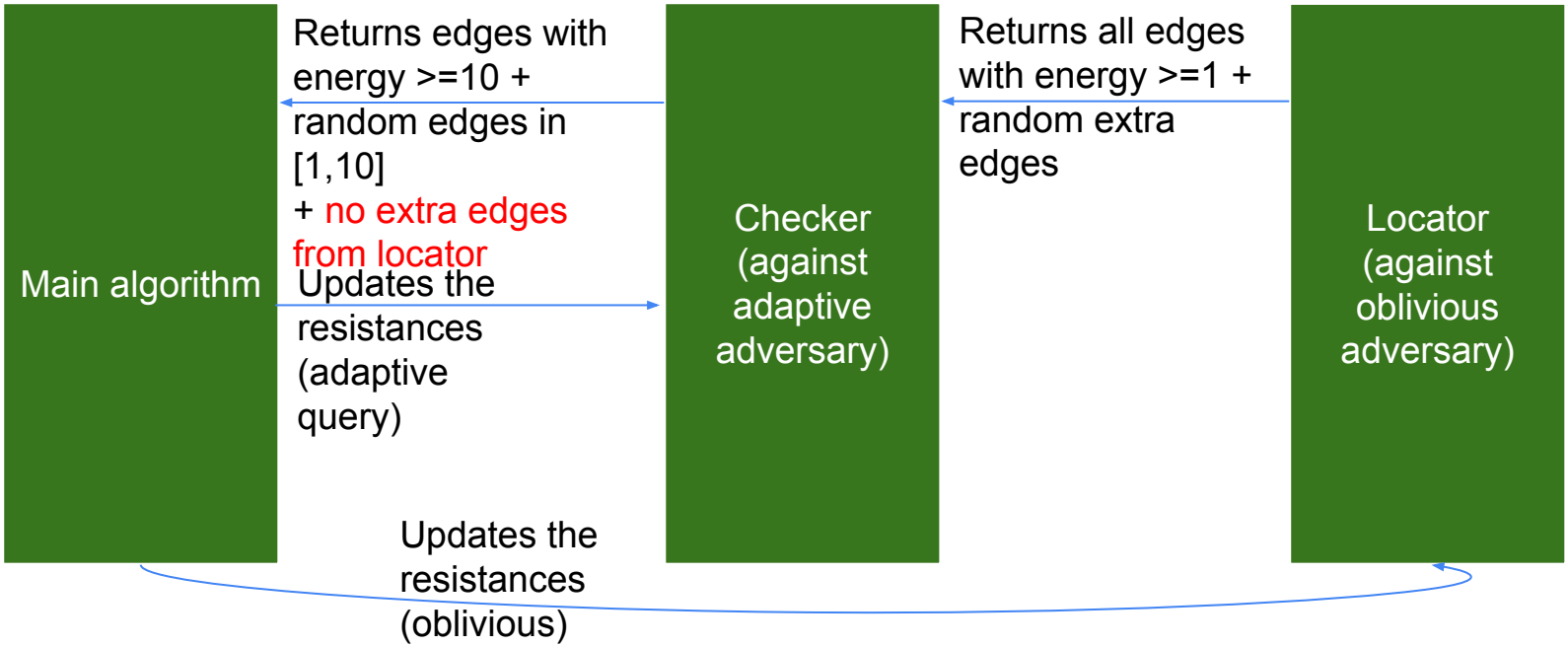
We fix this by wrapping it inside a separate algorithm *Checker* which independently estimates the flow on edges with a separate dynamic SC.

Need to carefully ensure that extra edges returned by the heavy hitter are not "accepted" by *Checker*, i.e. returned to the main IPM to update.

Also need to make sure extra edges returned by heavy hitter don't affect the responses of *Checker* on remaining edges.

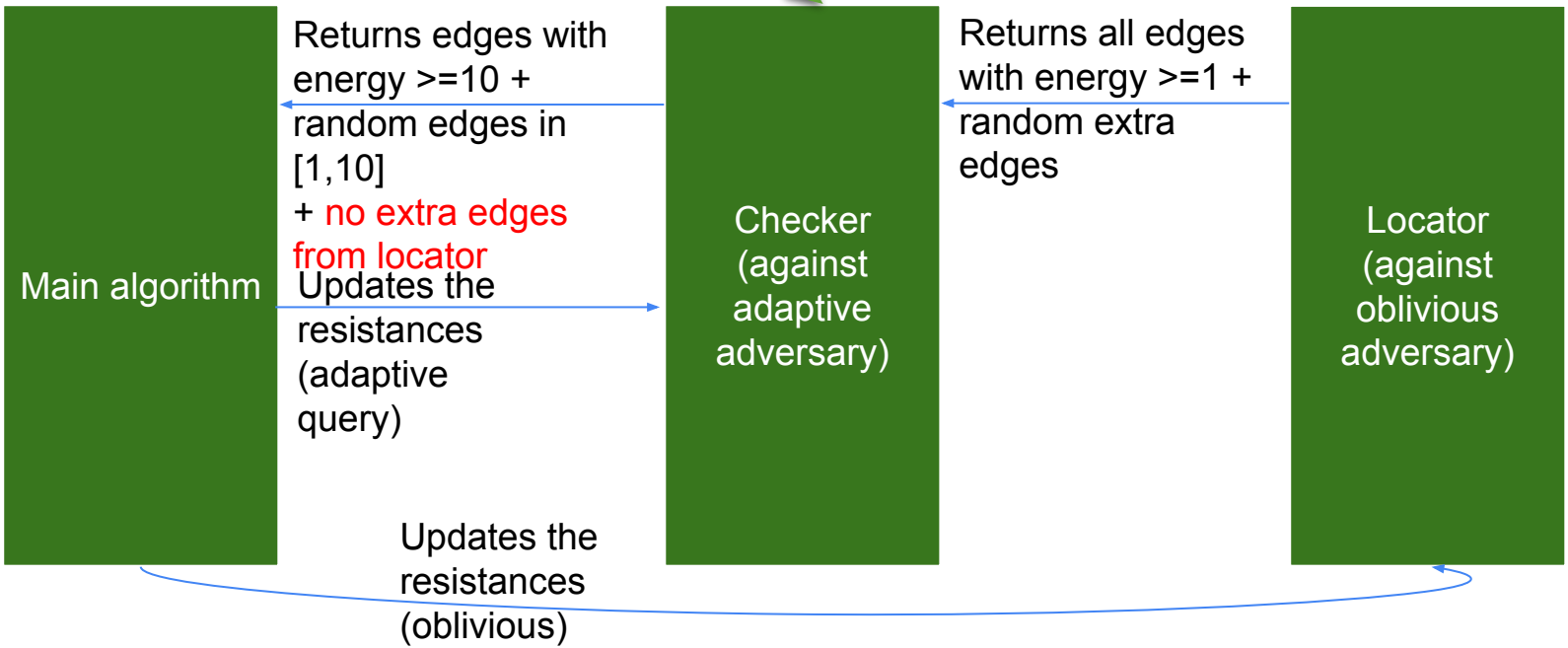

Main algorithm
Returns all edges
with energy >= 10
Borderline edges
are picked
randomly
Updates the
resistances
Adaptive query!
Locator

Adaptive *Checker*



Adaptive *Checker*

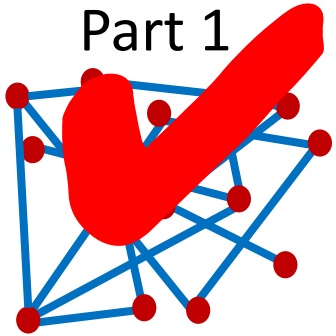
Make “semi-adaptive” by oversampling to capture all possible terminal sets / Schur complements possibly encountered.



Talk Organization

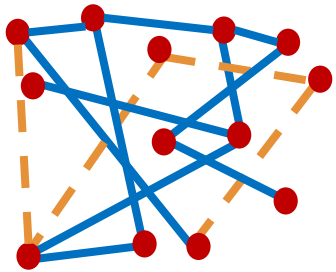
Flow Problems

Part 1



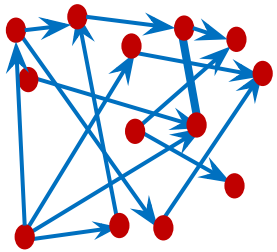
Fully Dynamic Electric Flows

Part 2



Putting it Together: Full Algorithm

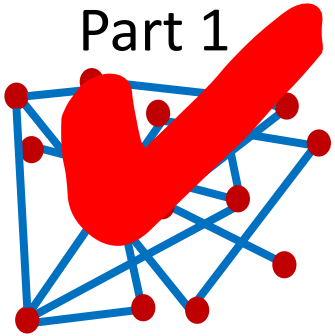
Part 3



Talk Organization

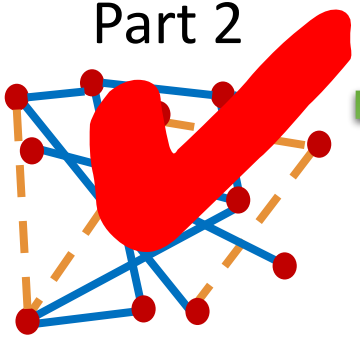
Flow Problems

Part 1



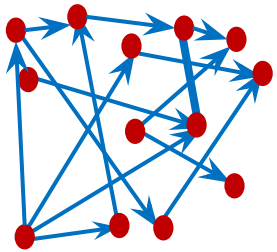
Fully Dynamic Electric Flows

Part 2



Putting it Together: Full Algorithm

Part 3



Full Algorithmic Picture

$k = m^{1/328}$. Goal is reduce residual flow by $(1-k/m^{1/2})$ in $\tilde{O}(m)$ amortized time.

Full Algorithmic Picture

$k = m^{1/328}$. Goal is reduce residual flow by $(1-k/m^{1/2})$ in $\tilde{O}(m)$ amortized time.

Initialize data structures: *Checker* and *Locator* (heavy hitter) to accuracy ε.

Full Algorithmic Picture

$k = m^{1/328}$. Goal is reduce residual flow by $(1-k/m^{1/2})$ in $\tilde{O}(m)$ amortized time.

Initialize data structures: *Checker* and *Locator* (heavy hitter) to accuracy ε.

For $\tilde{O}(m^{1/2}/k)$ batched steps:

Split into k^4 equally sized steps of size $1/(k^3 m^{1/2})$.

Locator + *Checker* return edges S with >= ε fraction of flow in s-t electric flow.

Update resistances of edges in S.

Full Algorithmic Picture

$k = m^{1/328}$. Goal is reduce residual flow by $(1-k/m^{1/2})$ in $\tilde{O}(m)$ amortized time.

Initialize data structures: *Checker* and *Locator* (heavy hitter) to accuracy ε.

For $\tilde{O}(m^{1/2}/k)$ batched steps:

Split into k^4 equally sized steps of size $1/(k^3 m^{1/2})$.

Locator + *Checker* return edges S with >= ε fraction of flow in s-t electric flow.

Update resistances of edges in S, pass to *Checker* and *Locator*.

Pay $\tilde{O}(m)$ time to bring flow back to “central path” of the interior point method.

Heuristic Analysis

$k = m^{1/328}$. Goal is reduce residual flow by $(1-k/m^{1/2})$ in $\tilde{O}(m)$ amortized time.

Initialize data structures: *Checker* and *Locator* (heavy hitter) to accuracy ε.

For $\tilde{O}(m^{1/2}/k)$ batched steps:

Split into k^4 equally sized steps of size $1/(k^3 m^{1/2})$.

Locator + *Checker* return edges S with >= ε fraction of flow in s-t electric flow.

Update resistances of edges in S, pass to *Checker* and *Locator*.

Pay $\tilde{O}(m)$ time to bring flow back to “central path” of the interior point method.

Heuristic Analysis

$k = m^{1/328}$. Goal is reduce residual flow by $(1-k/m^{1/2})$ in $\tilde{O}(m)$ amortized time.

Initialize data structures: *Checker* and *Locator* (heavy hitter) to accuracy ε.

For $\tilde{O}(m^{1/2}/k)$ batched steps:

Split into k^4 equally sized steps of size $1/(k^3 m^{1/2})$.

Locator + *Checker* return edges S with >= ε fraction of flow in s-t electric flow.

Update resistances of edges in S, pass to C

$k^4\varepsilon^{-2} = O(k^{16})$ updates.

Pay $\tilde{O}(m)$ time to bring flow back to "central path" of the interior point method.

Heuristic Analysis

$k = m^{1/328}$. Goal is reduce residual flow by $(1-k/m^{1/2})$ in $\tilde{O}(m)$ amortized time.

Initialize data structures: *Checker* and *Locator* (heavy hitter) to accuracy ε.

For $\tilde{O}(m^{1/2}/k)$ batched steps:

Amortized m^{1-c} per call / update.

Split into k^4 equally sized steps of size $1/(k^3 m^{1/2})$.

Locator + *Checker* return edges S with >= ε fraction of flow in s-t electric flow.

Update resistances of edges in S, pass to C

$k^4\varepsilon^{-2} = O(k^{16})$ updates.

Pay $\tilde{O}(m)$ time to bring flow back to "central path" of the interior point method.

Heuristic Analysis

Pay $\tilde{O}(m)$ after each "batched step" x $\tilde{O}(m^{1/2}/k) = \tilde{O}(m^{3/2}/k)$.

$O(k^{16})$ updates per phase x $\tilde{O}(m^{1/2}/k)$ phases x $O(m^{1-c})$ per update = $\tilde{O}(m^{3/2-c}k^{15})$.

Heuristic Analysis

Pay $\tilde{O}(m)$ after each “batched step” x $\tilde{O}(m^{1/2}/k) = \tilde{O}(m^{3/2}/k)$.

$O(k^{16})$ updates per phase x $\tilde{O}(m^{1/2}/k)$ phases x $O(m^{1-c})$ per update = $\tilde{O}(m^{3/2-c}k^{15})$.

$\tilde{O}(m^{3/2}/k) = \tilde{O}(m^{3/2-c}k^{15})$ -> $k = m^{c/16}$.

Total runtime is $\tilde{O}(m^{3/2-c/16})$ << $m^{1.5}$ at least!

Heuristic Analysis

Pay $\tilde{O}(m)$ after each “batched step” x $\tilde{O}(m^{1/2}/k) = \tilde{O}(m^{3/2}/k)$.

$O(k^{16})$ updates per phase x $\tilde{O}(m^{1/2}/k)$ phases x $O(m^{1-c})$ per update = $\tilde{O}(m^{3/2-c}k^{15})$.

$\tilde{O}(m^{3/2}/k) = \tilde{O}(m^{3/2-c}k^{15})$ -> $k = m^{c/16}$.

Total runtime is $\tilde{O}(m^{3/2-c/16}) << m^{1.5}$ at least!

(log U) dependence comes from capacity scaling: our algorithm can handle arbitrary polynomial sized weights in $\tilde{O}(m^{3/2-1/328})$.

Additional Difficulties

Analysis of the interior point method is slightly non-standard, as we only use s-t electric flows during the method.

Standard IPM adds in electric circulations between steps to “recenter” a little.

Additional Difficulties

Analysis of the interior point method is slightly non-standard, as we only use s-t electric flows during the method.

Standard IPM adds in electric circulations between steps to “recenter” a little.

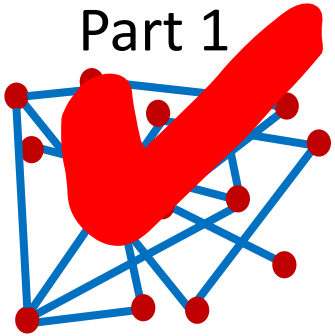
Some resistance updates accumulate over the course of several electric flow updates (instead of during a batched step).

Need to update such edges and trade off with cost of resistance updates in *Checker* and *Locator*.

Talk Organization

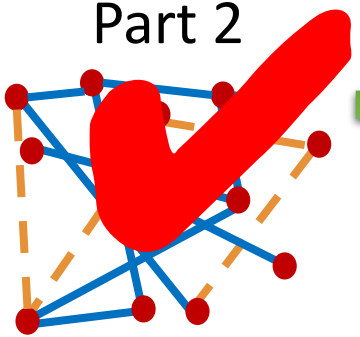
Flow Problems

Part 1



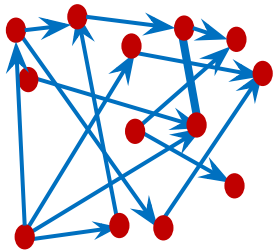
Fully Dynamic Electric Flows

Part 2



Putting it Together: Full Algorithm

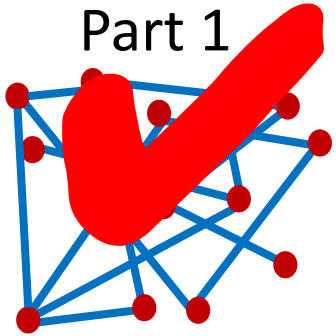
Part 3



Talk Organization

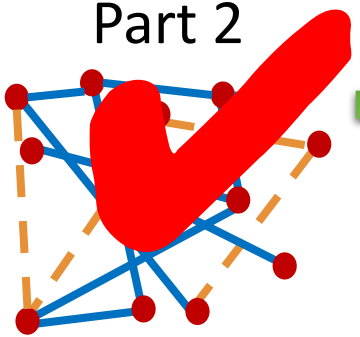
Flow Problems

Part 1



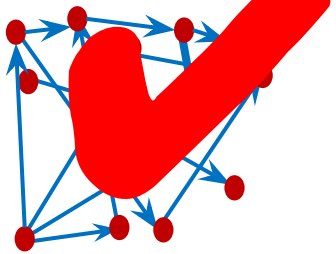
Fully Dynamic Electric Flows

Part 2



Putting it Together: Full Algorithm

Part 3



Conclusion

We give an algorithm that solves maxflow in $\tilde{O}(m^{1.497} \log U)$ on capacitated graphs

First improvement to the $\tilde{O}(m^{1.5} \log U)$ algorithm of Goldberg-Rao 1998.

Conclusion

We give an algorithm that solves maxflow in $\tilde{O}(m^{1.497} \log U)$ on capacitated graphs

First improvement to the $\tilde{O}(m^{1.5} \log U)$ algorithm of Goldberg-Rao 1998.

Approach is based on augmenting electrical flows and sublinear m^{1-c} algorithm for a specific instance of a dynamic electric flow problem.

Precisely, detect and return high energy edges in the s-t electric flow on a dynamic graph with changing resistances.

Further Directions / Open Problems

We believe that every piece of our algorithm can be improved or simplified.

Further Directions / Open Problems

We believe that every piece of our algorithm can be improved or simplified.

Outer method (IPM): Current IPM is highly tailored to the data structures, in particular that we primarily use s-t flows (instead of more general demands).

Standard IPM uses general demands / circulations (and better parameters).

Further Directions / Open Problems

We believe that every piece of our algorithm can be improved or simplified.

Outer method (IPM): Current IPM is highly tailored to the data structures, in particular that we primarily use s-t flows (instead of more general demands).

Standard IPM uses general demands / circulations (and better parameters).

Heavy hitter: Large dependencies on parameters, only slightly sublinear. Only handles s-t flows. Improve $(1+\varepsilon)$-approximate dynamic ER in general?

Further Directions / Open Problems

We believe that every piece of our algorithm can be improved or simplified.

Outer method (IPM): Current IPM is highly tailored to the data structures, in particular that we primarily use s-t flows (instead of more general demands).

Standard IPM uses general demands / circulations (and better parameters).

Heavy hitter: Large dependencies on parameters, only slightly sublinear. Only handles s-t flows. Improve (1+ε)-approximate dynamic ER in general?

Adaptivity: Current solution involves using a Checker and oversampling to control *all possible* terminal sets. Resparsification? [**BBGNSSS20**]

Fully Dynamic Electrical Flows:
Sparse Maxflow Faster than Goldberg-Rao

arXiv: 2101.07233

The End

Questions?

Yang P. Liu

Contact Info:
- email: *yangpliu@stanford.edu*
- website: *yangpliu.github.io*