Computing Flows in Subquadratic Space

Jan van den Brand
Georgia Tech

Zhao Song
Independent

Albert Weng
Georgia Tech

Problem Statement

Minimum Cost Flow

Given a directed graph G,
find flow $x \in \mathbb{R}^E$ that

- minimizes total cost along edges $c^\top x$

while it

- routes vertex demands d,
- subject to edge capacities u.

Generalizes max st flow.

Linear Program

$$\begin{aligned} \min\ & c^\top x \\ s.t.\ \ & A^\top x = d \\ & 0 \le x \le u \end{aligned}$$

Space Complexity

focus

Important in streaming, parallel/distributed computing, communication complexity, etc.
Goal: flows in *subquadratic* space

naturally extends

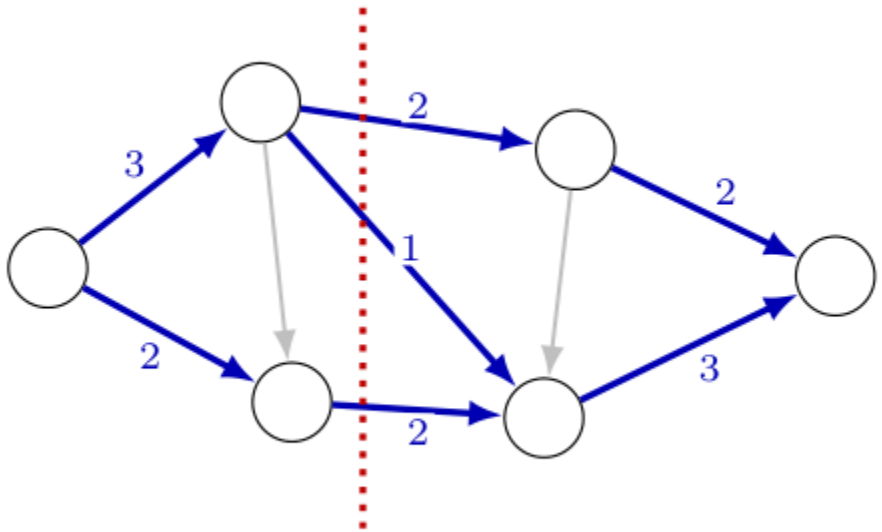
Subquadratic in number of vertices. Not obvious, since there could be quadratically many edges!

Model of Computation - Streaming

- We cannot store the entire graph.
- Input (i.e., edge set) is given to us one-by-one as a stream of data.
- *Multi-pass streaming*: algorithm allowed multiple passes through input.

Well studied for bipartite matching but not flow.

Flow in a Streaming Setting



e_1 (v_1, v_2) c_1 u_1	e_2	e_3	…	e_m (m can be $\Omega(n^2)$)

Simpler Problem: Flow size

Return size of max flow after end of passes through stream

- [CJWY25]: $(1+\epsilon)$-approximate with $\tilde{O}(n)$ space and 1 pass
- [RSW18]: undirected exact flow size (and min cut) with $\tilde{O}(n^{5/3})$ space

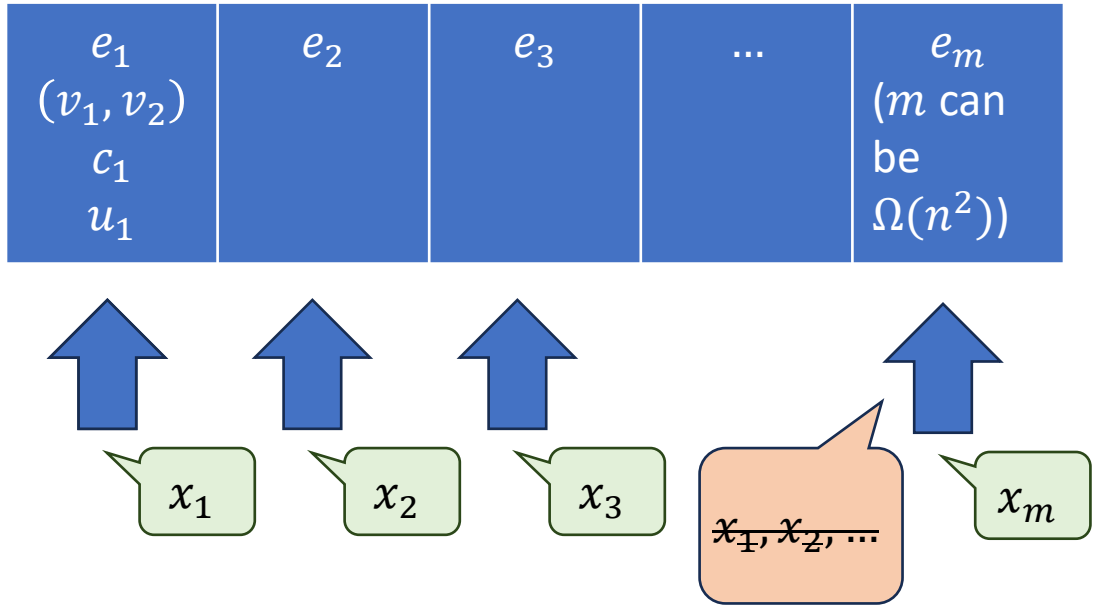
Our Problem: Flow in every edge

Seems difficult without being able to store the graph!

[CJWY25]: Impossible in $o(n^2)$ space

Slight Model Change

[CJWY25]: Impossible to encode/store approximate max flow in $o(n^2)$ space.



Our Result: you can both represent and compute flows in $O\left(n^{1.5} \log 1/\epsilon\right)$ space, using $\tilde{O}\left(\sqrt{n}\right)$ passes through the stream.

How we circumvent this lower bound:

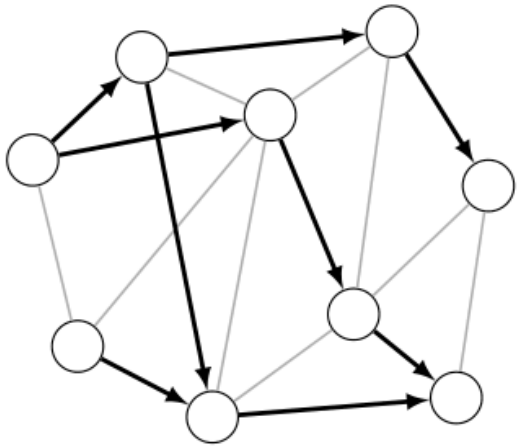
- one extra pass through stream to output the flow

first nontrivial upper bound

Computing Flows in Subquadratic Space

Augmenting Electric Circulations

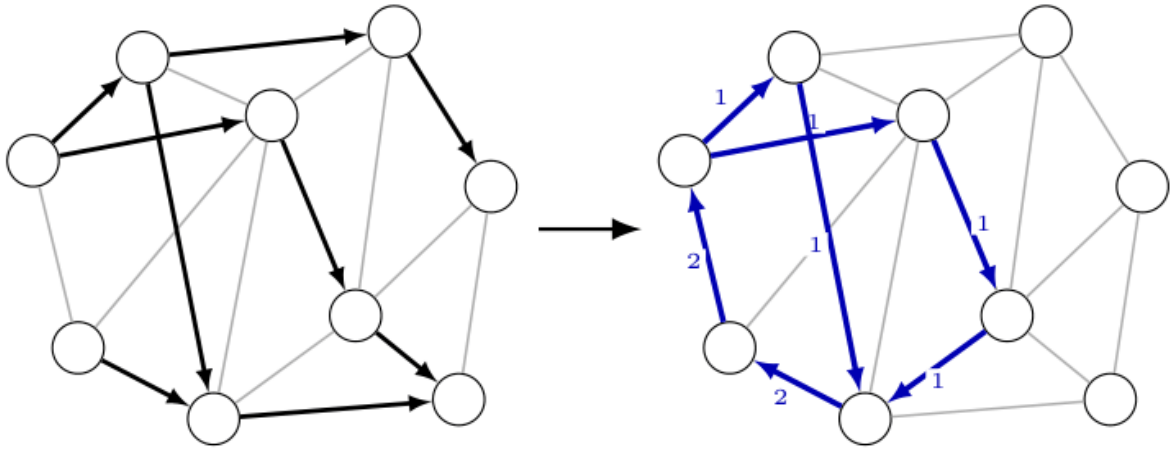
Min cost flow = original feasible flow + many augmenting circulations



start with some feasible flow

Augmenting Electric Circulations

Min cost flow = original feasible flow + many augmenting circulations



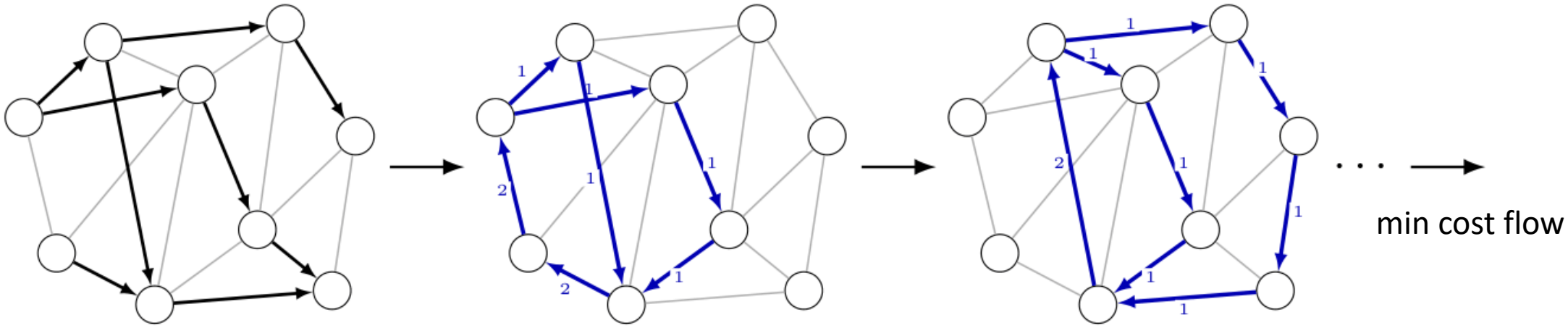
start with some feasible flow

solve an *electric circulation* problem

- depends on edge costs and current flow
- 0 net demand across vertices (circulation)

Augmenting Electric Circulations

$\sqrt{n}$ circulations suffice



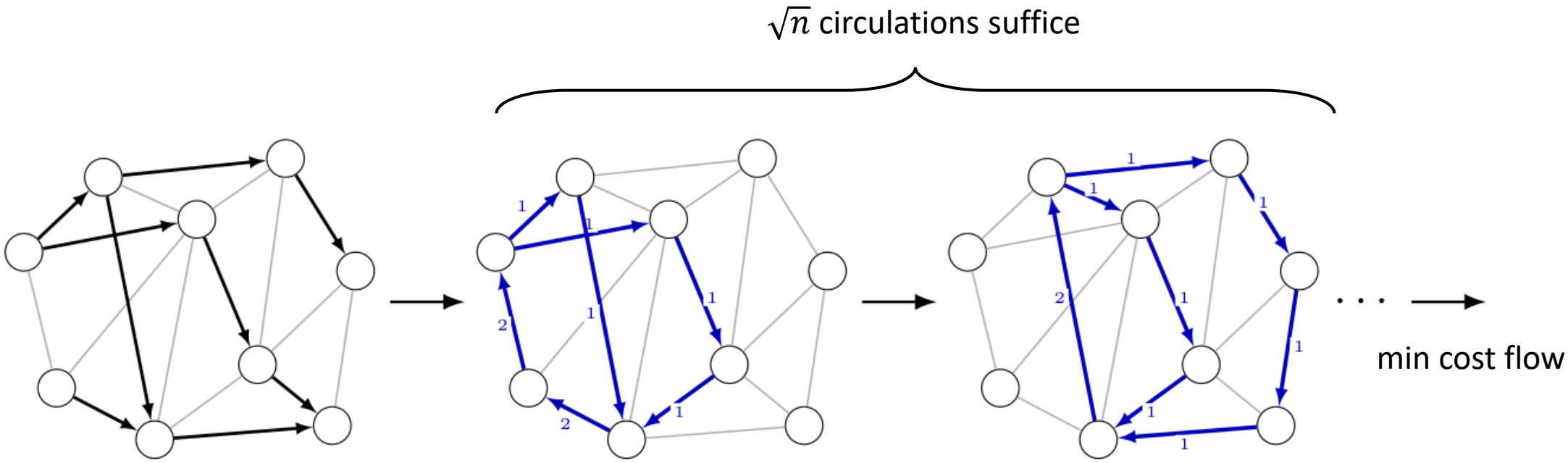
start with some feasible flow

solve an *electric circulation* problem

- depends on edge costs and current flow
- 0 net demand across vertices (circulation)

min cost flow

Space Complexity Analysis

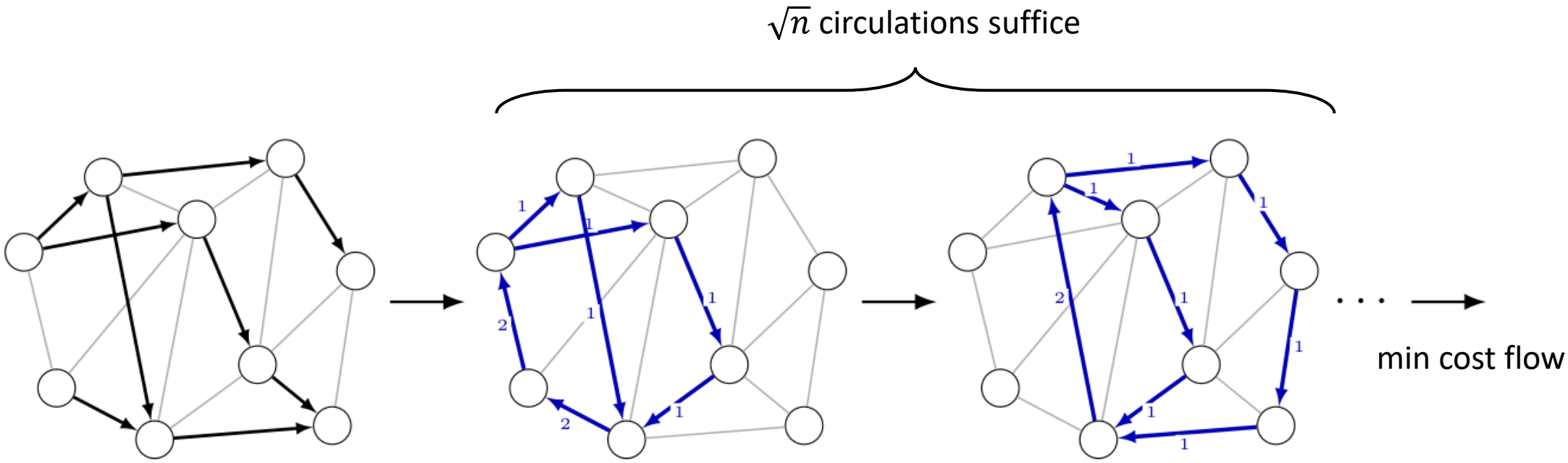


start with some feasible flow
$O(n)$ space
(some flexibility in choosing)

solve an *electric circulation* problem

- depends on edge costs and current flow
- 0 net demand across vertices (circulation)

Space Complexity Analysis



start with some feasible flow
$O(n)$ space
(some flexibility in choosing)

solve an *electric circulation* problem

- depends on edge costs and current flow
- 0 net demand across vertices (circulation)
- store the solution in $O(n)$ space

Storing Electric Circulations

Electric circulations:

$$f + g,$$

the sum of

- "gradient" flow g that reduces cost
- electric flow f that fixes demands

Can be compressed to $\tilde{O}(1)$ space (details omitted)

Can be represented by vertex potentials and resistances:

for $e = (u, v)$

$$f_e = \frac{\phi_u - \phi_v}{r_e}.$$

- ϕ: store explicitly in $O(n)$ space
- r: depends mainly on amount of flow x_e already on this edge. Use previous saved potentials and backtrack!

Storing Resistances and Lewis Weights

To ensure $\sqrt{n}$ circulations we need to choose resistances r carefully:

$$r_e = \tau_e \cdot \left(\frac{1}{x_e^2} - \frac{1}{(u_e - x_e)^2} \right)$$

where

- u_e is the capacity
- x_e is the current flow
- τ_e is the Lewis weight of edge e

$$x_e = x_e^{(0)} + (g_e^{(1)} + f_e^{(1)}) + (g_e^{(2)} + f_e^{(2)}) + \cdots$$

Backtrack to find previous f_e, then the one before that, etc.

ℓ_p Lewis weight: "importance" of an edge

$$\tau_e = \sigma\left(\tau^{\frac{1}{2}-\frac{1}{p}} \left(\frac{1}{x^2} - \frac{1}{(u-x)^2} \right) A \right)_e + \frac{n}{m}$$

where $\sigma(M)$ is the leverage scores of M:

$$\sigma(M)_e = (M(M^\top M)^{-1} M^\top)_{e,e}.$$

Storing Lewis Weights

[ST04]: Definition and sketching leverage scores of M:

$$\sigma(M)_e = (M(M^\top M)^{-1}M^\top)_{e,e} = ||M(M^\top M)^{-1}M^\top \chi_e||_2^2$$
$$\approx ||RM(M^\top M)^{-1}M^\top \chi_e||_2^2.$$

where R is a log-dimension JL sketch matrix.

Start with $\vec{1}$.
Then store $\tilde{O}(1)$ JL sketches of the leverage scores.

Store in $\tilde{O}(1)$ space!

ℓ_p Lewis weight
(we use $p = 1 - 1/\Theta(\log m)$)

$$\tau_e = \sigma\left(\tau^{\frac{1}{2}-\frac{1}{p}}\left(\frac{1}{x^2} - \frac{1}{(u-x)^2}\right)A\right)_e + \frac{n}{m}.$$

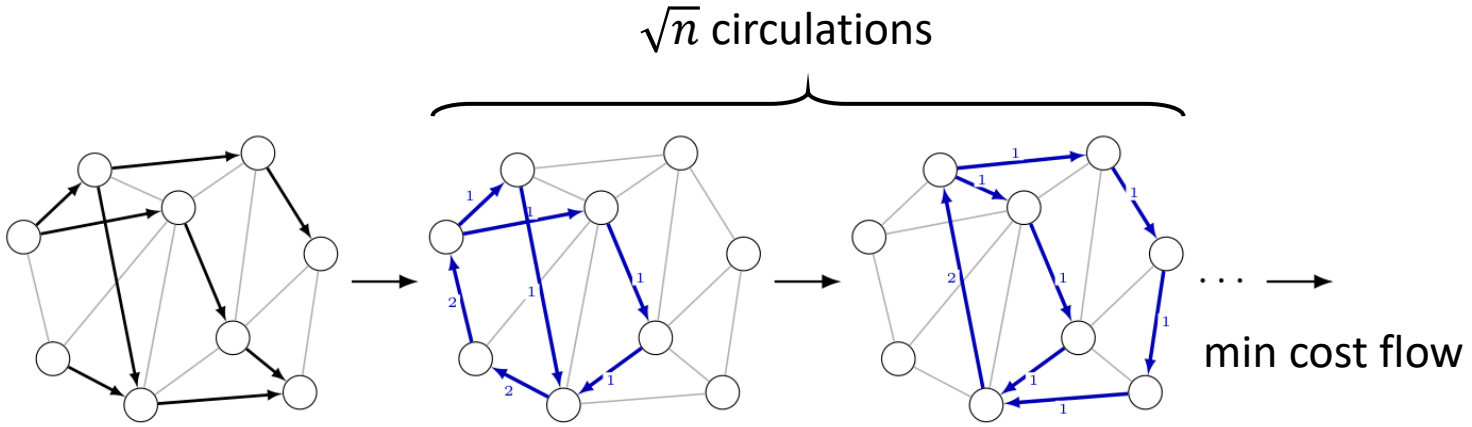
Note that this definition is recursive!

[CP15]: Suffices to start with some (possibly bad) approximation w, then iteratively refine:

$$w \leftarrow \left(w^{2/p-1}(\sigma(\cdot) + n/m)\right)^{p/2}$$

Converges in $\tilde{O}(1)$ iterations.

Summary



- Store starting feasible flow: $O(n)$ space
- Store $O(\sqrt{n})$ electrical flows. Each flow consists of
  - Vertex potentials: $O(n)$ space
  - Edge resistances
    - τ_e: Store JL sketches for weights in $O(1)$ space
    - x_e: Backtrack through previous flows
- Store $O(\sqrt{n})$ gradients g: $O(1)$ space each

Conclusion

Summary

- First subquadratic space calculation of flows
- Circumvents flow lower bounds by returning output during the last read-through of the input
- First subquadratic communication complexity algorithm for flow.

Open

- Further reduce space complexity (is $O(n)$ possible?)
- Reduce number of passes