Electrical Flows, Optimization, and New Approaches to the Maximum Flow Problem

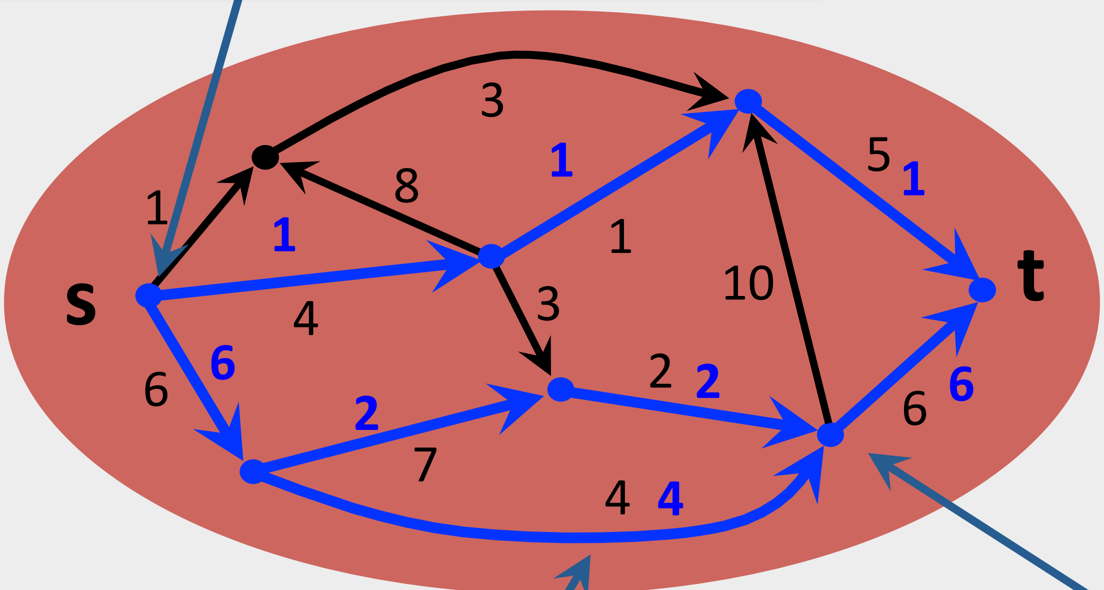
Aleksander Mądry

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Maximum flow problem

Input: Directed graph **G**, integer **capacities u_e**, **source s** and **sink t**

value = net flow out of **s**



Here, value = **7**

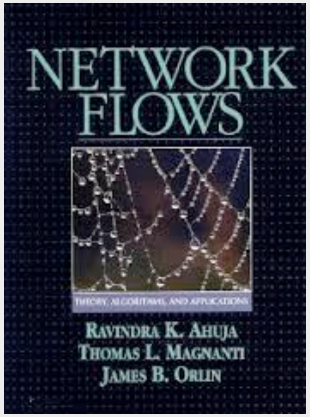
no overflow on arcs:
$0 \le f(e) \le u(e)$

no leaks at all **v≠s,t**

Task: Find a **feasible s-t flow** of **max value**

What is known about Max Flow?

A (very) rough history outline



[Dantzig '51]	$O(mn^2 U)$
[Ford Fulkerson '56]	$O(mn\, U)$
[Dinitz '70]	$O(mn^2)$
[Dinitz '70] [Edmonds Karp '72]	$O(m^2 n)$
[Dinitz '73] [Edmonds Karp '72]	$O(m^2 \log U)$
[Dinitz '73] [Gabow '85]	$O(mn \log U)$
[Goldberg Rao '98]	$\tilde{O}(m \min(m^{1/2}, n^{2/3}) \log U)$
[Lee Sidford '14]	$\tilde{O}(mn^{1/2} \log U)$

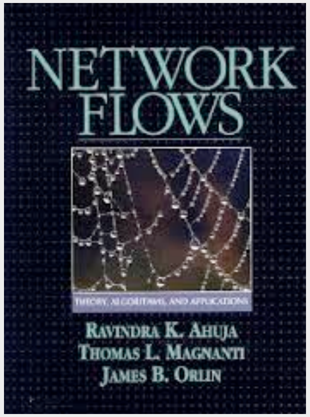
Our focus: Sparse graph (**m=O(n)**) and unit-capacity (**U=1**) regime

→ It is a good benchmark for combinatorial graph algorithms
→ Already captures interesting problems, e.g., **bipartite matching**

(**n** = # of vertices, **m** = # of arcs, **U** = max capacity, **Õ()** hides polylogs)

What is known about Max Flow?

A (very) rough history outline



[Dantzig '51]	**O(n^3)**
[Ford Fulkerson '56]	**O(n^2)**
[Dinitz '70]	**O(n^3)**
[Dinitz '70] [Edmonds Karp '72]	**O(n^3)**
[Dinitz '73] [Edmonds Karp '72]	**Õ(n^2)**
[Dinitz '73] [Gabow '85]	**Õ(n^2)**
[Goldberg Rao '98]	**Õ($n^{3/2}$)**
[Lee Sidford '14]	**Õ($n^{3/2}$)**

Our focus: Sparse graph (**m=O(n)**) and unit-capacity (**U=1**) regime

→ It is a good benchmark for combinatorial graph algorithms
→ Already captures interesting problems, e.g., **bipartite matching**

(**n** = # of vertices, **m** = # of arcs, **U** = max capacity, **Õ()** hides polylogs)

Breaking the $O(n^{3/2})$ barrier

Undirected graphs and approx. answers ($O(n^{3/2})$ barrier still holds here)

[M '10]: **Crude approx. of** max flow **value** in **close to linear** time

[CKMST '11]: **(1-ε)-approx.** to max flow in $\tilde{O}(n^{4/3}\varepsilon^{-3})$ time

[LSR '13, S '13, KLOS '14]: **(1-ε)-approx.** in **close to linear** time

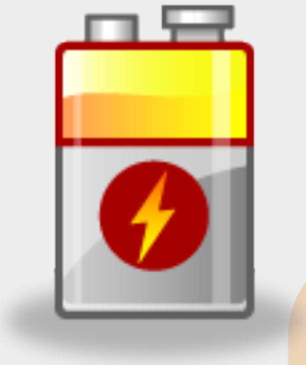
But: What about the **directed** and **exact** setting?

Today

[M '13]: Exact $\tilde{O}(n^{10/7}) = \tilde{O}(n^{1.43})$-time alg.

(n = # of vertices, $\tilde{O}()$ hides polylog factors)

Electrical flows



Input: **Undirected** graph **G**, **resistances** r_e, **source s** and **sink t**

Principle of least energy

Electrical flow of value F:
The unique minimizer of the **energy**

$$\mathbf{E(f) = \Sigma_e\, r_e\, f(e)^2}$$

among all **s-t** flows **f** of value **F**

Electrical flows = ℓ_2-minimization

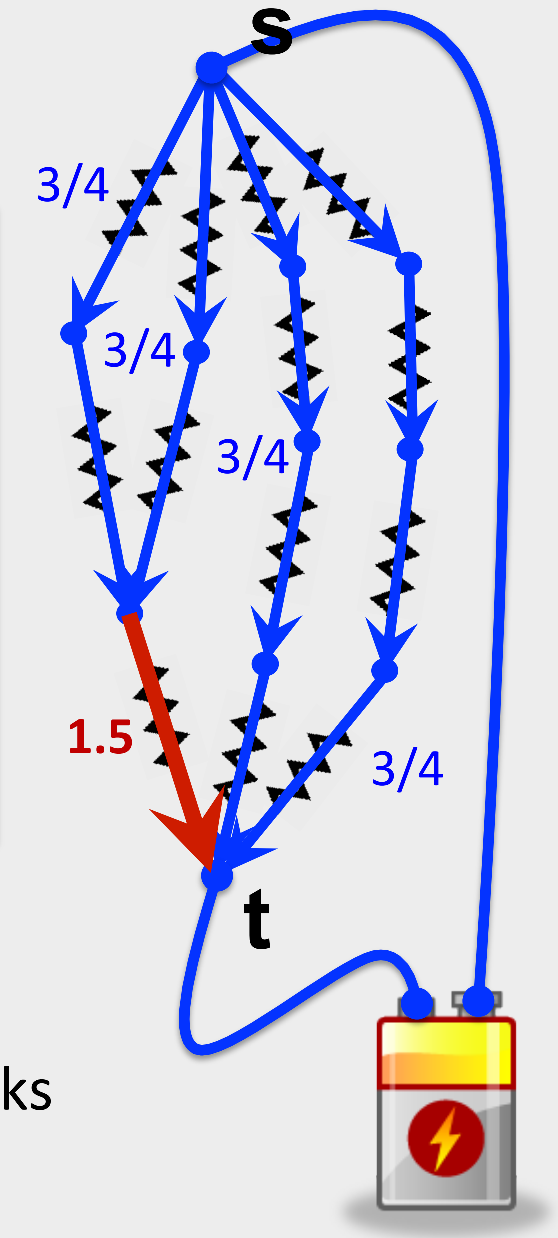
Recall: We can compute it in **nearly-linear** time

From electrical flows to undirected max flow

Approx. undirected max flow [Christiano Kelner M. Spielman Teng '11] via electrical flows

Assume: $\mathbf{F}^*$ known (via binary search)

→ Treat edges as resistors of resistance **1**
→ Compute electrical flow of value $\mathbf{F}^*$
(This flow has **no leaks**, but **can overflow** some edges)
→ To fix that: **Increase resistances** on the overflowing edges
Repeat (**hope:** it doesn't happen too often)



Surprisingly: This approach can be made work!

But: One needs to be careful how to fill in the blanks

We will do this now

Filling in the blanks

Recall: We are dealing with **undirected** graphs

From now on: All capacities are **1**, **m=O(n)**
and the value $\mathbf{F}^*$ of max flow is known

Electrical vs. maximum flows

Fix some resistances **r** and consider the elect. flow $\mathbf{f_E}$ of value **F***

We don't expect $\mathbf{f_E}$ to obey **all** capacity constraints
(i.e., we can have $|\mathbf{f_E(e)}| \gg \mathbf{1}$ for some edge **e**)

Still, $\mathbf{f_E}$ obeys these constraints in a certain sense...

We have:

$$\Sigma_e\, r_e\, |f_E(e)| \leq \Sigma_e\, r_e$$

In other words: Capacity constraints are preserved on **average** (weighted wrt to $\mathbf{r_e}$s)

Electrical vs. maximum flows

This gives rise to a **very fast** algorithm for the following task:

'Feasibility on average':

Given weights **w** compute a flow **f** of value **F*** s.t.

$$\Sigma_e w_e |f(e)| \leq \Sigma_e w_e$$

Key point: We already know how to make such a crude algorithm useful to us!

Multiplicative weights update method
[FS ’97, PST ’95, AHK ’05]

‘Technique for turning weak algorithms into strong ones’

In our setting:

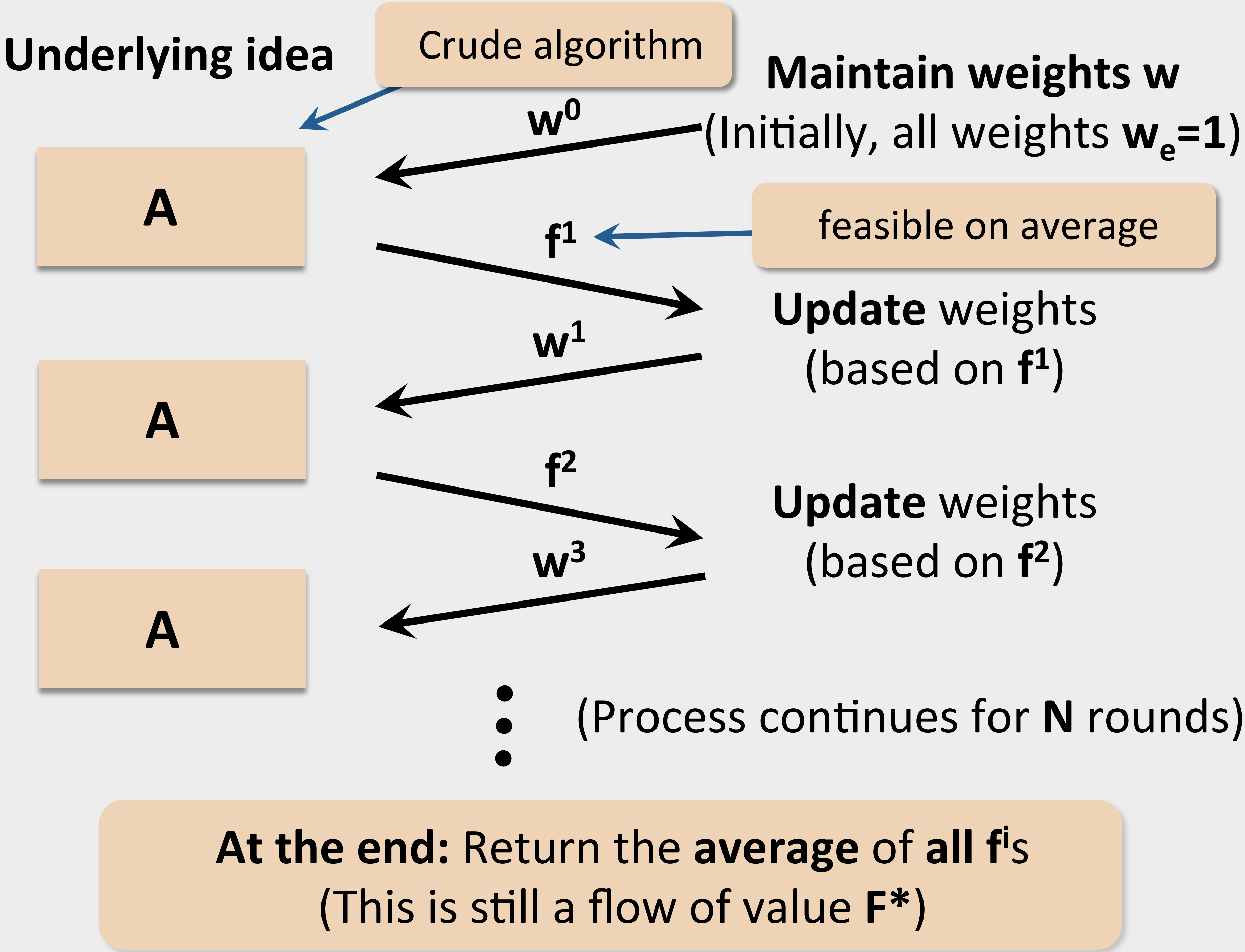
Crude algorithm computing **‘feasible on average’** flows

↓

(1-ε)-approx. max flow

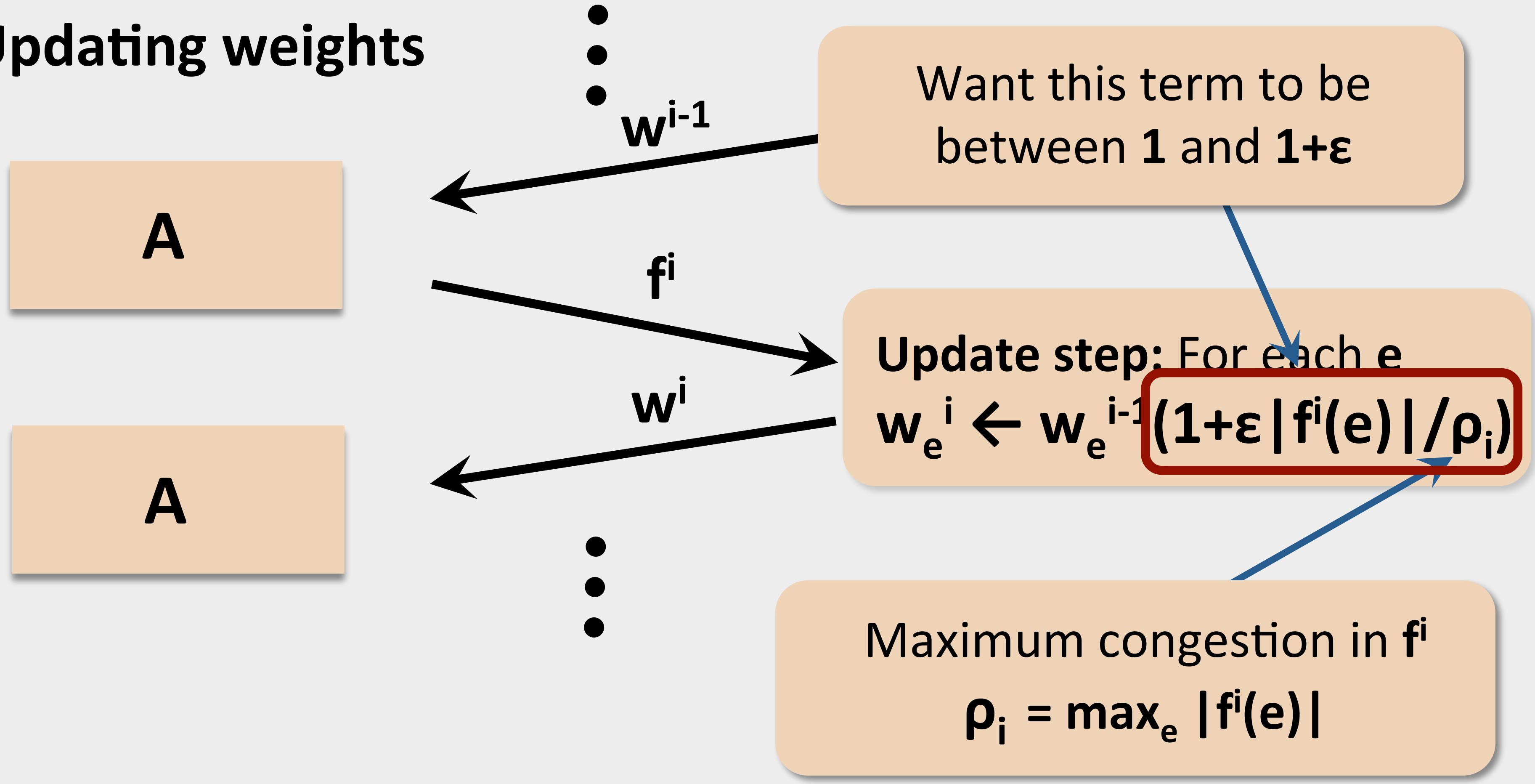
[**(1+ε)-approx**. feasibility **everywhere**]

How does this method work?

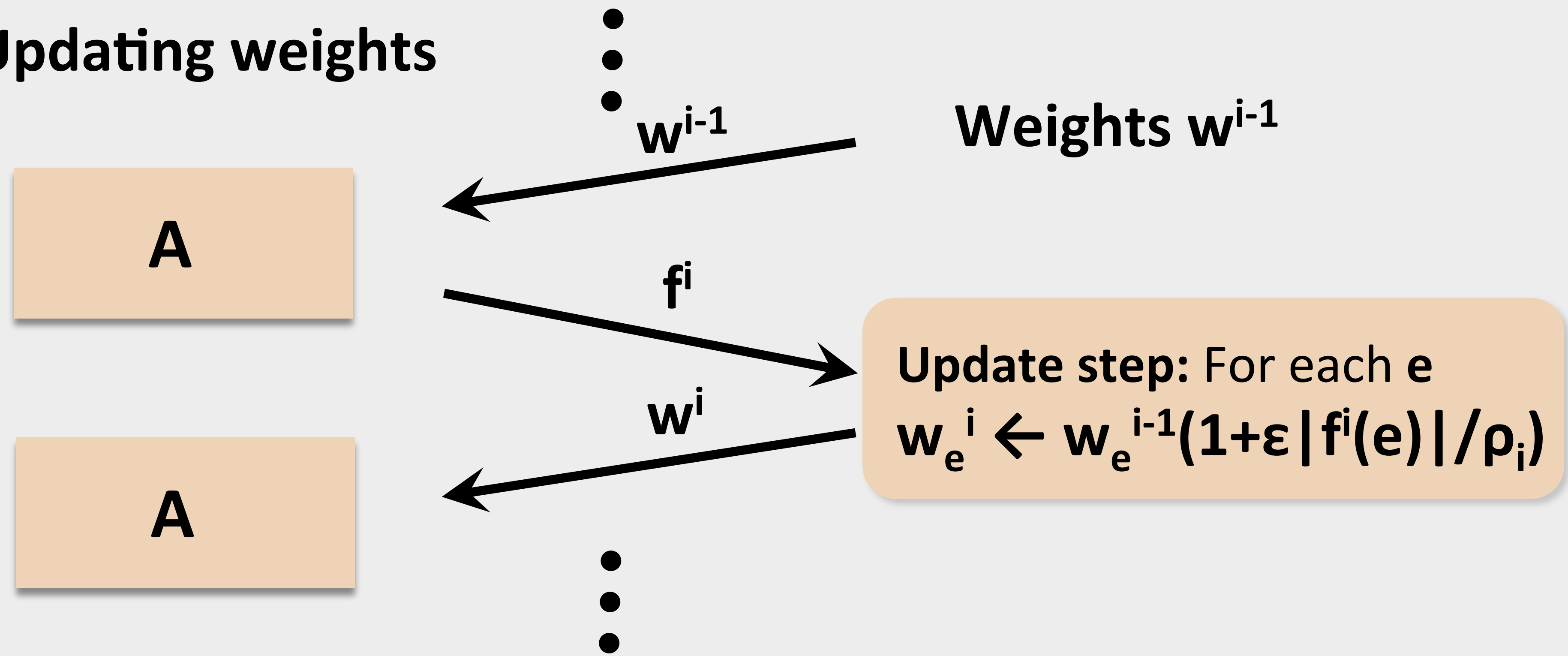


Underlying idea
Crude algorithm
Maintain weights w
(Initially, all weights $\mathbf{w_e=1}$)
$\mathbf{w^0}$
A
$\mathbf{f^1}$
feasible on average
Update weights
(based on $\mathbf{f^1}$)
$\mathbf{w^1}$
A
$\mathbf{f^2}$
Update weights
(based on $\mathbf{f^2}$)
$\mathbf{w^3}$
A
(Process continues for **N** rounds)
At the end: Return the **average** of **all** $\mathbf{f^i}$s
(This is still a flow of value **F***)

Updating weights



Updating weights



Underlying dynamics:

Edge **e suffers large** overflow → $\mathbf{w_e}$ grows **rapidly**

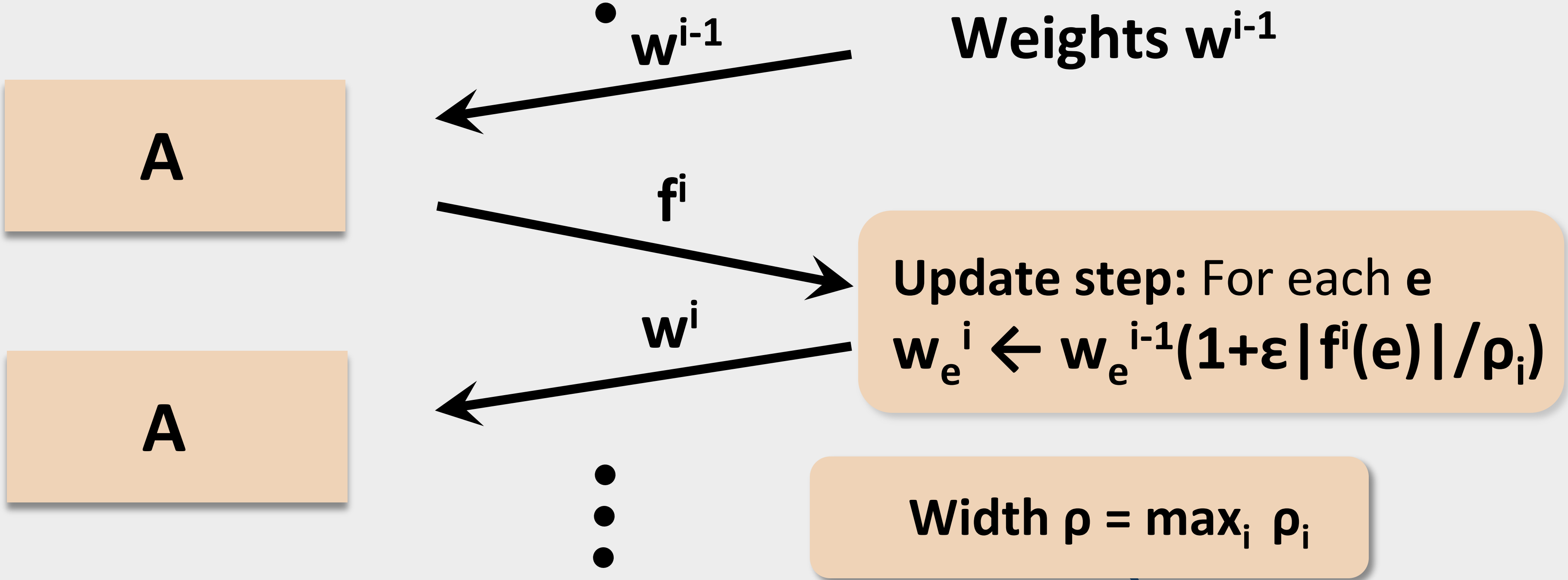
Average overflow **small** → $\mathbf{\Sigma_e w_e}$ grows **slowly**

↓

No edge suffers large overflow **too often**

→ **averaging out** yields (almost) no overflow

Updating weights



[AHK ’05]: It suffices to repeat this step $\mathbf{N=\tilde{O}(\rho\varepsilon^{-2})}$ times to get a **(1-ε)-approx** to max flow

Think: **ρ** measures the **electrical vs. max** flow discrepancy

Note: Linear dependence on **ρ** is unavoidable

Bottom line:

A

=

Electrical flow primitive gives us the crude algorithm

We can use MWU framework
to fill in our blanks!

Our algorithm

→ Treat edges as resistors of resistance $\mathbf{r_e=1}$

→ Compute electrical flow **f** of value $\mathbf{F^*}$

→ **Increase resistances** on overflowing edges

Repeat

Our algorithm

→ Treat edges as resistors of resistance $\mathbf{r_e=1}$

→ Compute electrical flow $\mathbf{f}$ of value $\mathbf{F^*}$

→ **Increase resistances:** for each **e**,

$$r_e^i \leftarrow r_e^{i-1}(1+\varepsilon|f^i(e)|/\rho_i)$$

Repeat $\mathbf{N=\tilde{O}(\rho\varepsilon^{-2})}$ **times**

→ **At the end:** Take an **average** of all the flows as the final answer

Resistances $\mathbf{r_e}$ evolve as weights $\mathbf{w_e}$

Convergence condition: “execute **N** rounds”

Our algorithm

→ Treat edges as resistors of resistance $\mathbf{r_e=1}$

→ Compute electrical flow **f** of value $\mathbf{F^*}$

→ **Increase resistances:** for each **e**,

$$r_e^i \leftarrow r_e^{i-1}(1+\varepsilon|f^i(e)|/\rho_i)$$

Repeat **$N=\tilde{O}(\rho\varepsilon^{-2})$ times**

→ **At the end:** Take an **average** of all the flows as the final answer

Result: This algorithm gives us an **(1-ε)**-approx. max flow in $\tilde{O}(\rho\varepsilon^{-2})\cdot\tilde{O}(n) = \tilde{O}(n\rho\varepsilon^{-2})$ time

Crucial question: How large the worst-case overflow **ρ** can be?

Our question: Let **f** be an elect. flow of value $\mathbf{F^*}$ wrt resist. $\mathbf{r_e}$
How large $\boldsymbol{\rho} = \mathbf{max_e\ |f(e)|}$ can be?

In general: $\boldsymbol{\rho}$ can be very large
(**Think:** one edge having an extremely small resistance)

Fix: Regularize the resistances with a uniform distribution
$$\mathbf{r_e' \leftarrow r_e + \varepsilon\ |r|_1 / m}$$

Can show: $\boldsymbol{\rho}$ is bounded by $\mathbf{O(n^{1/2}\, \varepsilon^{-1})}$ then

This gives a **(1-ε)-approx.** $\mathbf{\tilde{O}(n^{3/2}\varepsilon^{-3})}$**-time** algorithm

Going beyond the $\tilde{O}(n^{3/2})$ Barrier

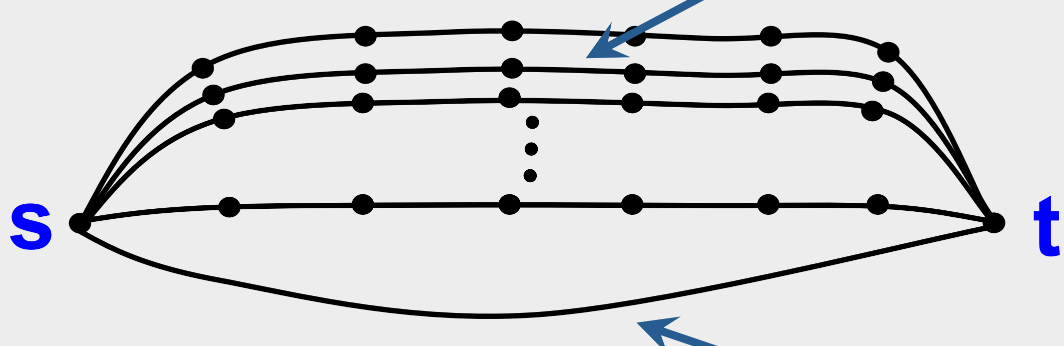
Speeding up our algorithm

Running time is dominated by **≈ρ** elect. flow computations

Can we improve our **O($n^{½}$ $ε^{-1}$)** bound on **ρ**?

Not really…

≈$n^{½}$ paths with **≈$n^{½}$** vertices each



one edge

Speeding up our algorithm

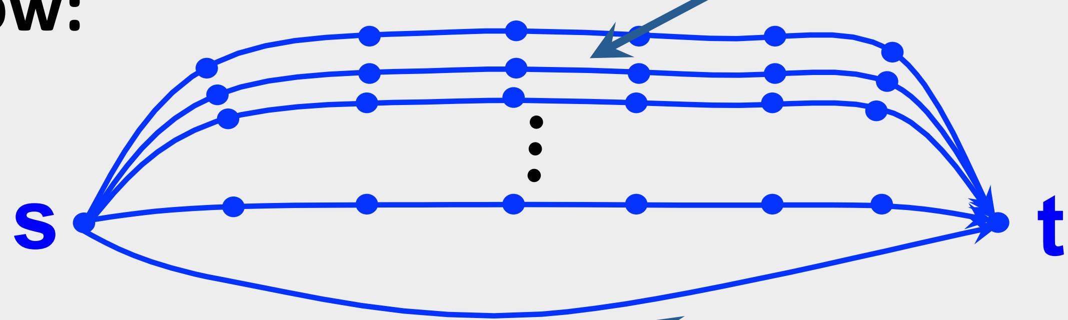
Running time is dominated by $\approx\boldsymbol{\rho}$ elect. flow computations

Can we improve our $\mathbf{O(n^{1/2}\,\varepsilon^{-1})}$ bound on $\boldsymbol{\rho}$?

Not really…

$\approx\mathbf{n^{1/2}}$ paths with $\approx\mathbf{n^{1/2}}$ vertices each

Max flow:



$\mathbf{F^{*}\approx n^{1/2}}$

one edge

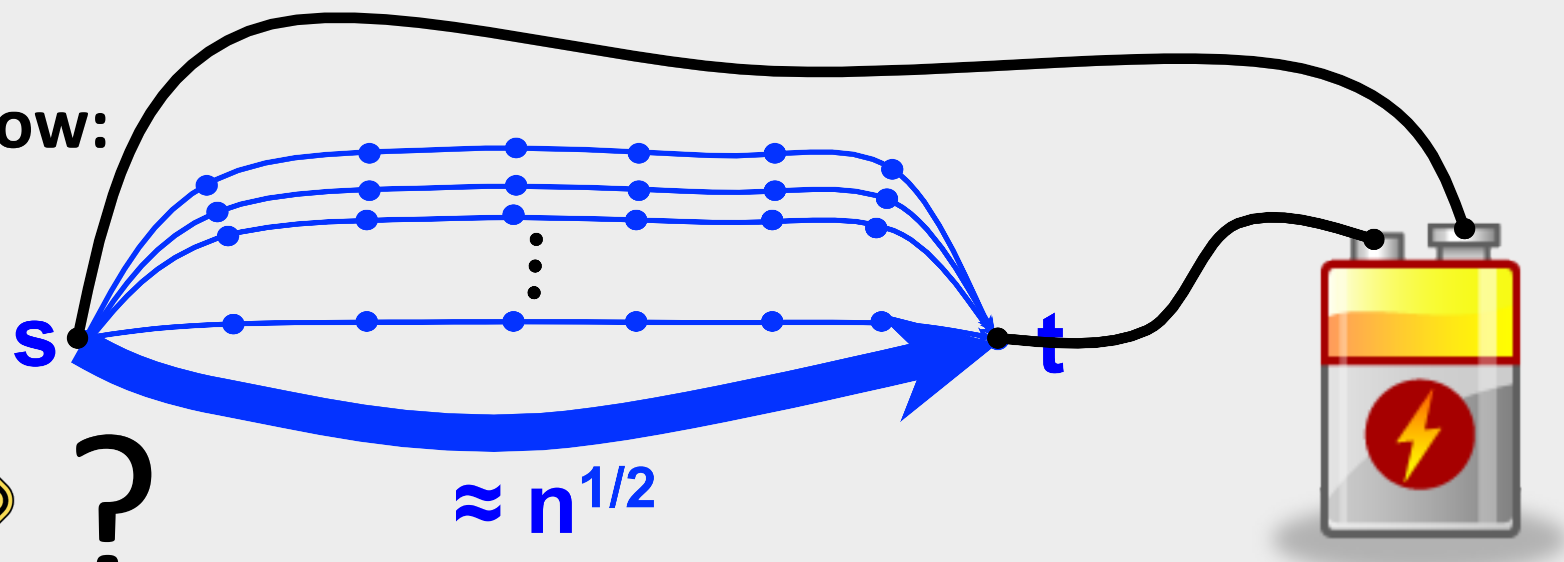
Speeding up our algorithm

Running time is dominated by $\approx\boldsymbol{\rho}$ elect. flow computations

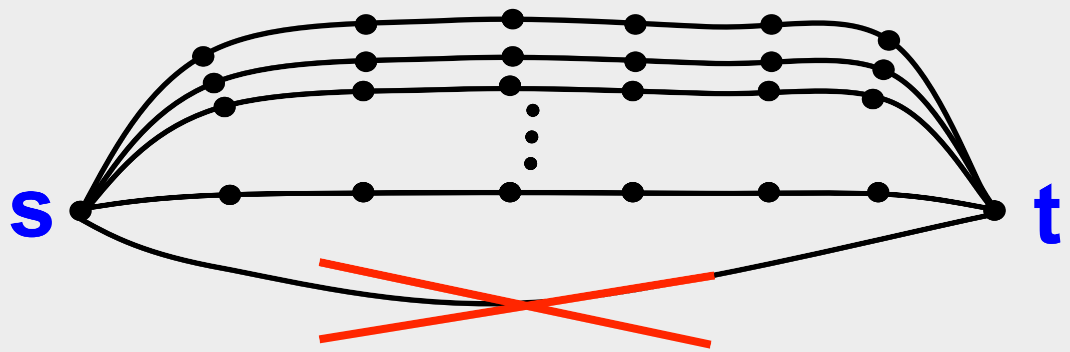
Can we improve our $\mathbf{O(n^{1/2}\,\varepsilon^{-1})}$ bound on $\boldsymbol{\rho}$?

Not really...

Electr. flow:



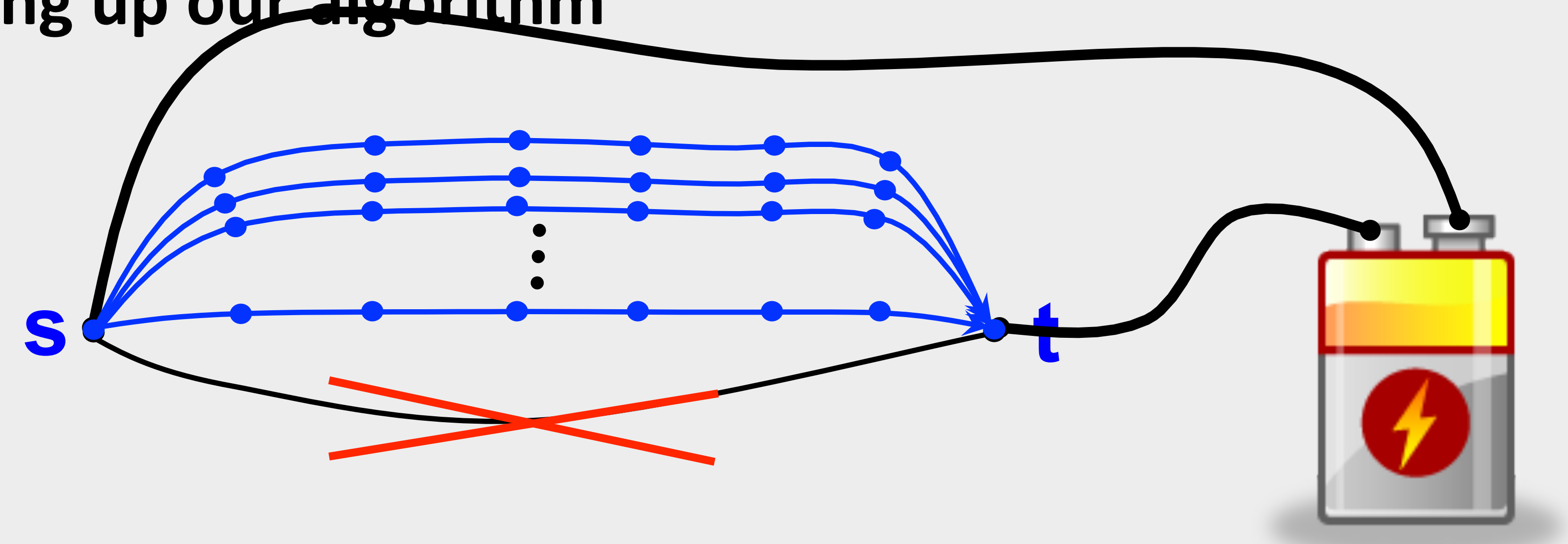
Speeding up our algorithm



Key observation: If we remove this bad edge…

→ The **max flow** does not change much

Speeding up our algorithm



Key observation: If we remove this bad edge…

→ The **max flow** does not change much

→ But the resulting **electrical flow** is much better behaved!

Can we turn this observation into an algorithmic idea?

Speeding up our algorithm

Idea: Let our electrical flow oracle **self-enforce** a smaller overflow $\boldsymbol{\rho' \ll \rho}$

Modification of the oracle: If the computed electrical flow has some edge **e** flow more than $\boldsymbol{\rho'}$:

→ **Remove** this edge from the graph (permanently)

→ **Recompute** the electrical flow

Note: If this oracle always **successfully** terminates, its effective overflow is $\boldsymbol{\rho'}$

Speeding up our algorithm

Crucial question: What is the right setting of **ρ'**?

→ We want **ρ'** to be as small as possible
→ But if it becomes too small the edge removal might be too aggressive and cut too many of them

Sweet spot: $\rho' \approx n^{1/3}$

Key reason: Removal of edges that flow a lot
→ significantly increases the **energy** of the electr. flow
→ But perturbs the **max flow** only slightly

Speeding up our algorithm

Our potential: The **energy $E_r(f)$** of the electrical flow **f** wrt current resistances **r**

Can show:

→ **$E_r(f)$** is not too small initially and cannot become too large (as long as we remove no more than **≈ εF^*** edges)

→ As the resistances only increase, **$E_r(f)$** never decreases

This makes $E_r(f)$ a convenient potential

Remaining piece: Removal of an overflowing edge increases **$E_r(f)$** significantly

This gives the **$\tilde{O}(n^{4/3}\varepsilon^{-3})$-time (1-ε)-approx.** algorithm

Thank you

Afternoon: Computing an exact max flow