Electrical Flows, Laplacian Matrices, and New Approaches to the Maximum Flow Problem

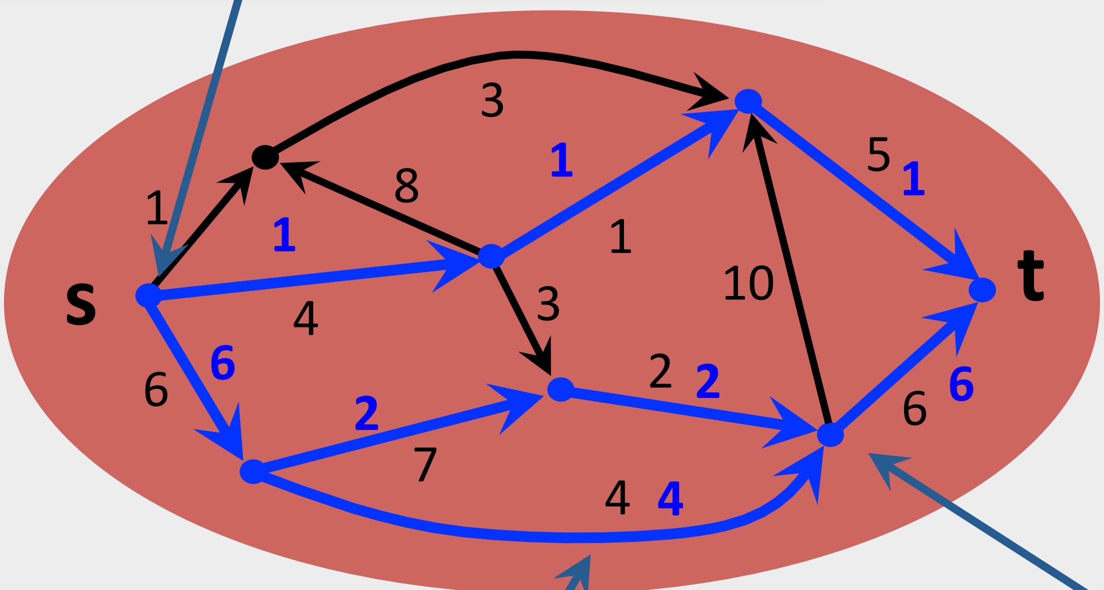
Aleksander Mądry

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 ≤ f(e) ≤ u(e)

no leaks at all **v≠s,t**

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

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?

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

This week

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

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

Today

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

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

This week

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

New approach:
Bring linear-algebraic techniques into play

Idea: Probe the **global flow structure** of the graph by **solving linear systems**

How to relate **flow structure** to **linear algebra**?
(And why should it even help?)

Key object: Electrical flows

Electrical flows

Input: **Undirected** graph **G**,
resistances $\mathbf{r_e}$,
source s and **sink t**

Principle of least energy

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

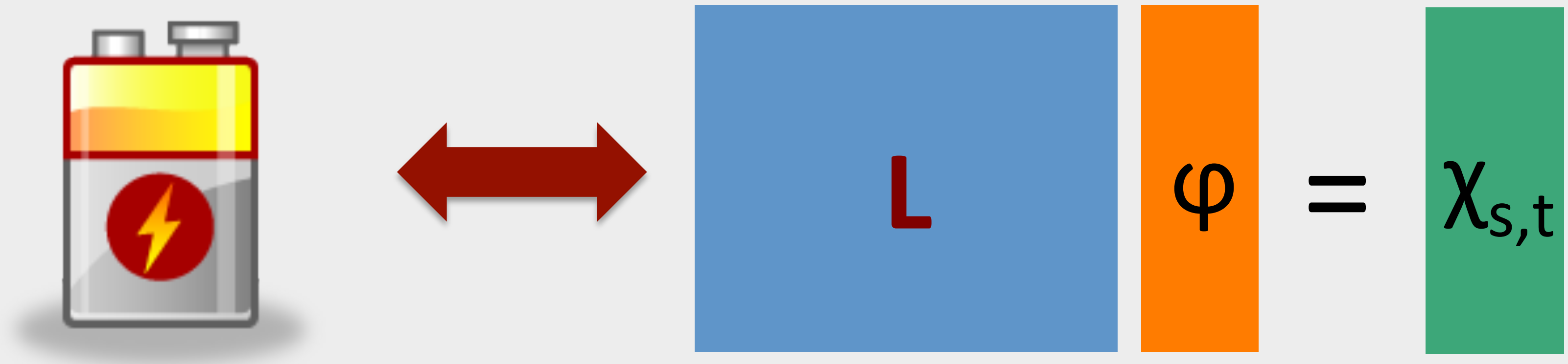
$$\mathbf{E(f) = \Sigma_e\, r_e\, f(e)^2}$$

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

Electrical flows = ℓ_2-minimization

How to compute an electrical flow?

Bottom line:



Electrical flow computation

Solving a **Laplacian system**

Bad news: Solving a linear system can take $\mathbf{O(n^{\omega})=O(n^{2.373})}$

(Prohibitive!)

Key observation:
$\mathbf{BR^{-1}B^{T}}$ is the **Laplacian** matrix **L**
of the underlying graph

How to utilize it?

Result: Electrical flow is a **nearly-linear time** primitive

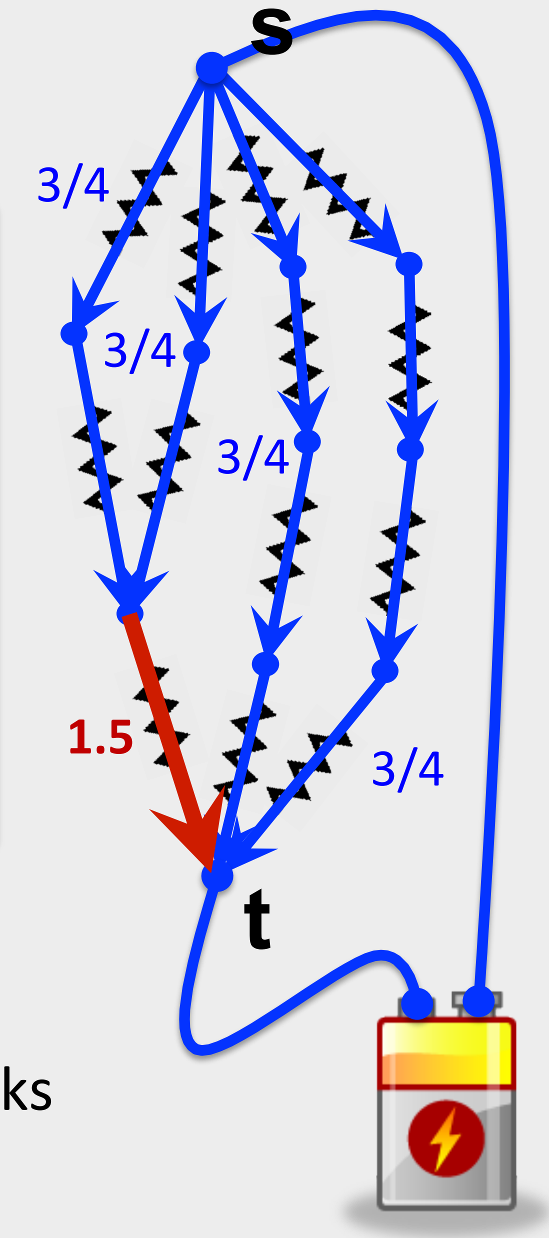
From electrical flows to undirected max flow

[CKMST '11]

Approx. undirected max flow via electrical flows

Assume: F^* known (via binary search)

→ Treat edges as resistors of resistance **1**
→ Compute electrical flow of value 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)

Proof:

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:

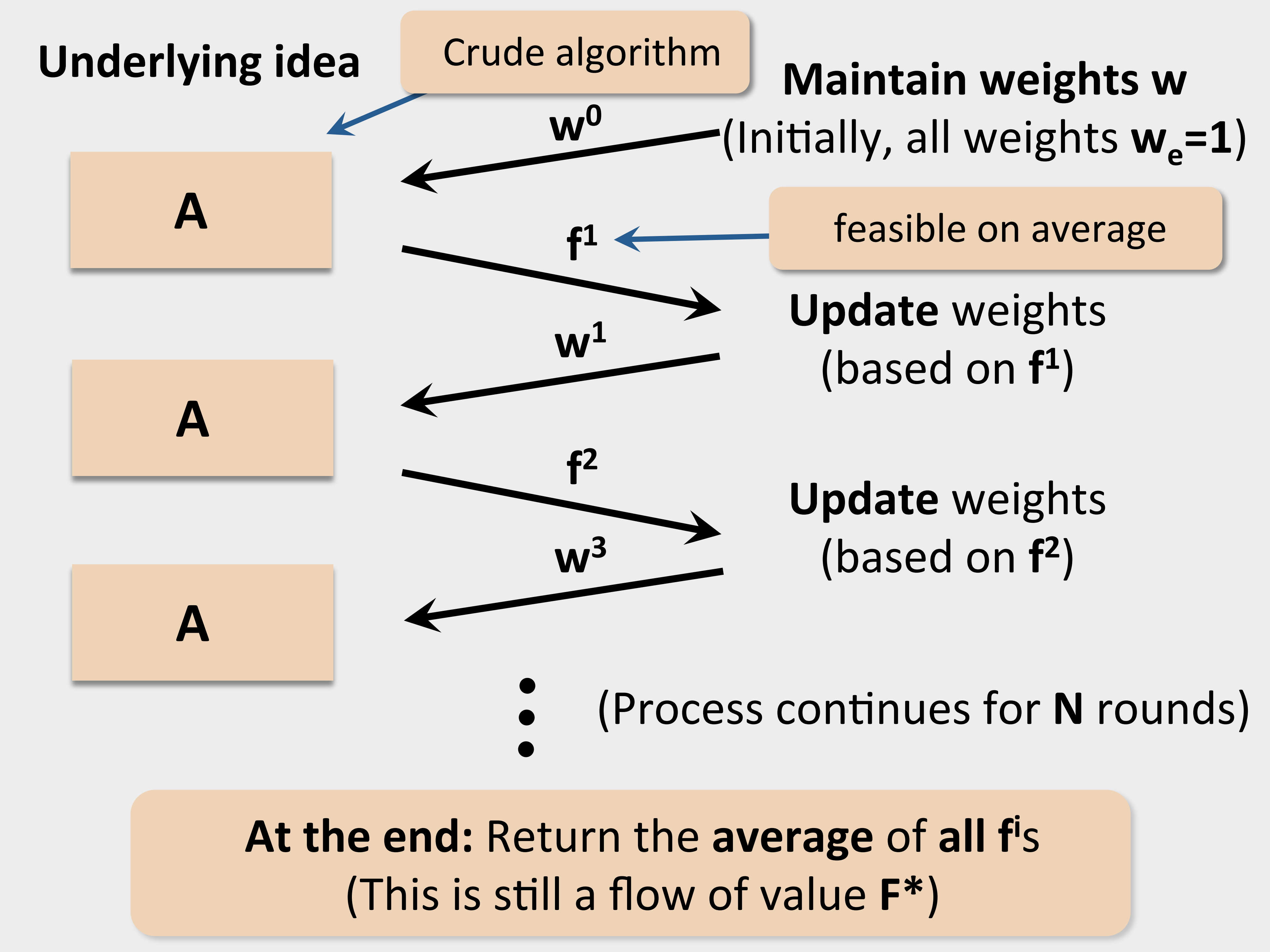
Crude algorithm computing **‘feasible on average’** flows

↓

(1-ε)-approx. max flow

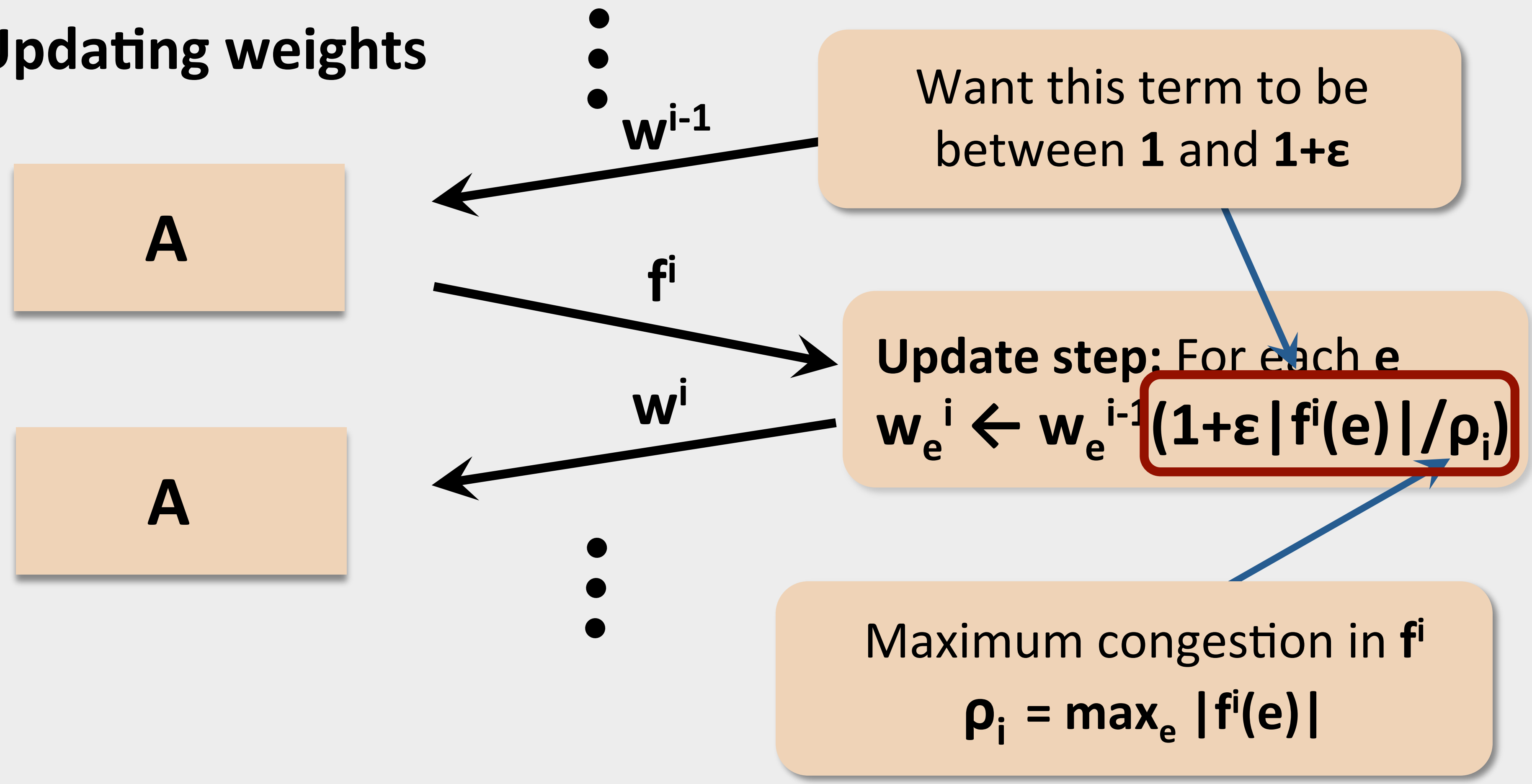
[**(1+ε)-approx**. feasibility **everywhere**]

How does this method work?

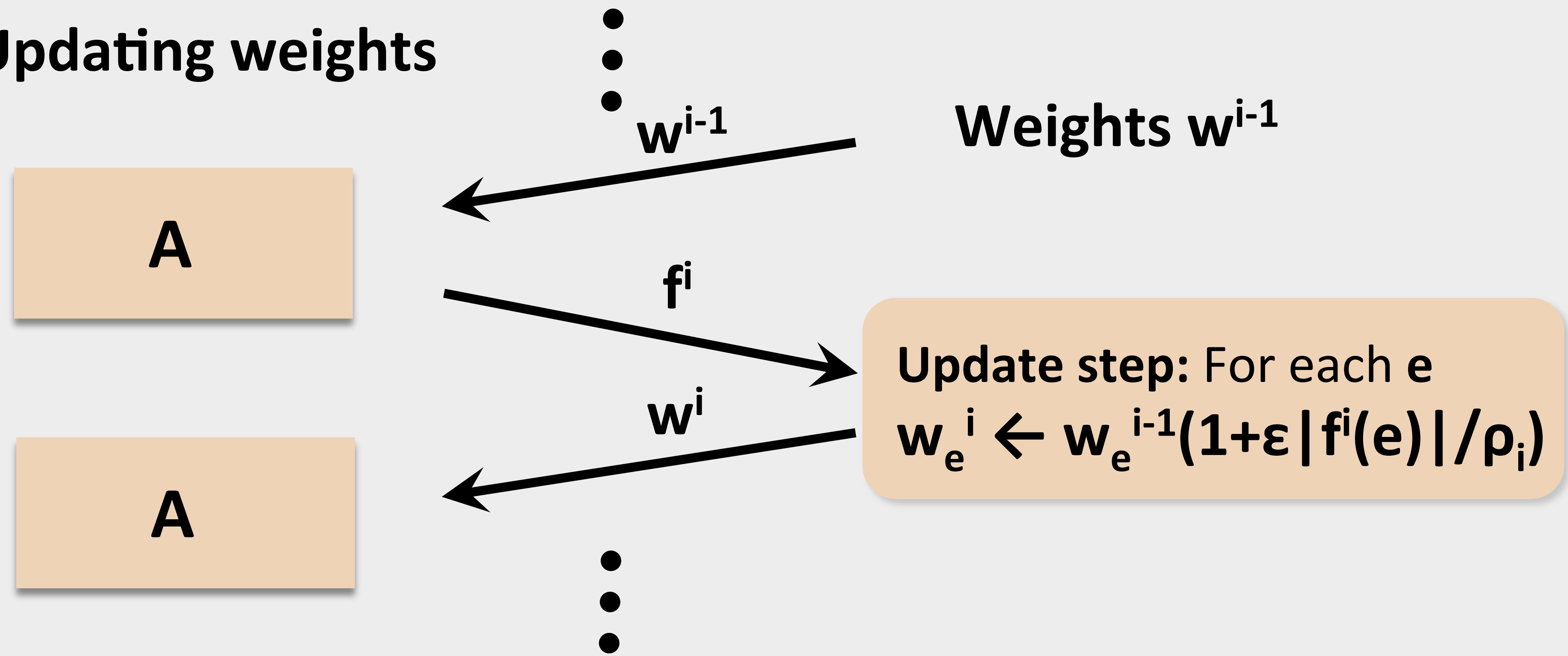


Underlying idea
Crude algorithm
Maintain weights w
(Initially, all weights w_e=1)
w^0
A
f^1
feasible on average
Update weights
(based on f^1)
w^1
A
f^2
Update weights
(based on f^2)
w^3
A
(Process continues for N rounds)
At the end: Return the average of all f^is
(This is still a flow of value F*)

Updating weights



Updating weights



Underlying dynamics:

Edge **e suffers large** overflow → w_e grows **rapidly**

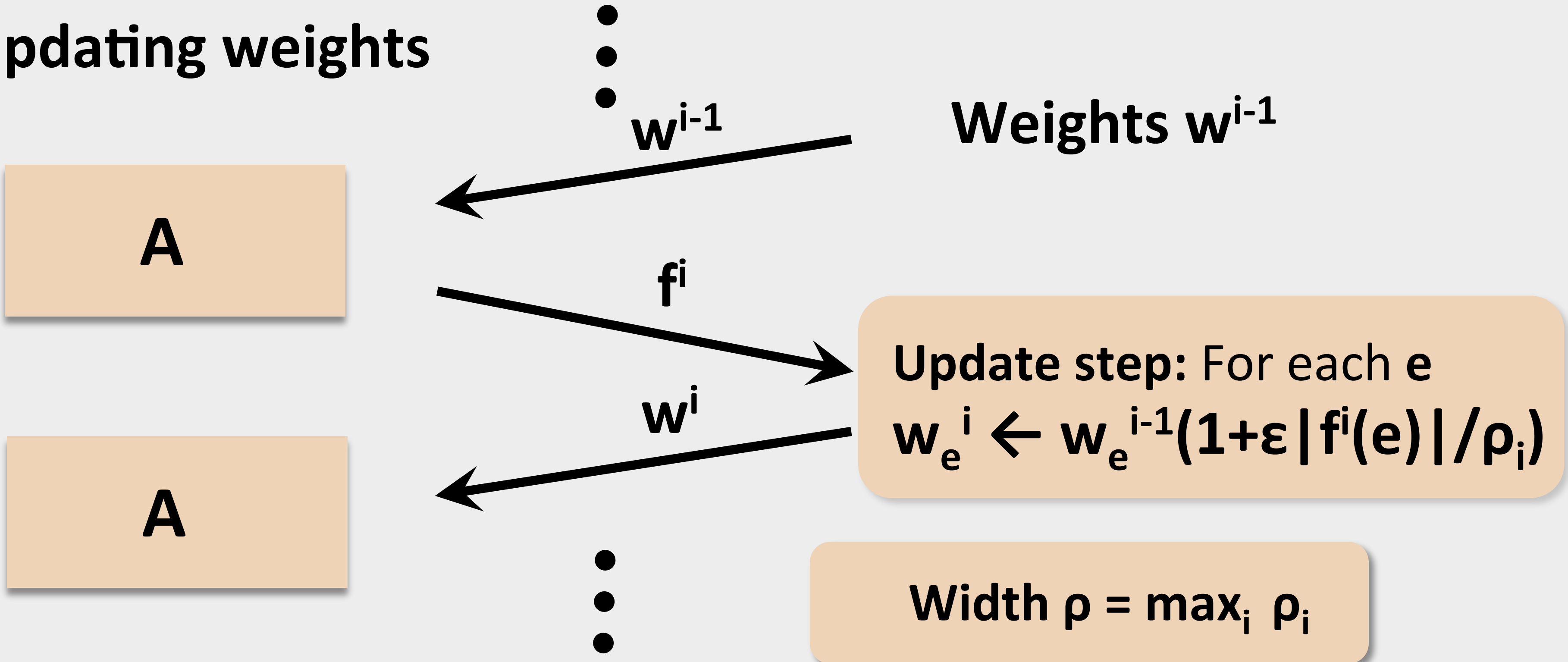
Average overflow **small** → $\Sigma_e w_e$ grows **slowly**

↓

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

Updating weights

Weights w^{i-1}

w^{i-1}

A

f^i

Update step: For each **e**

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

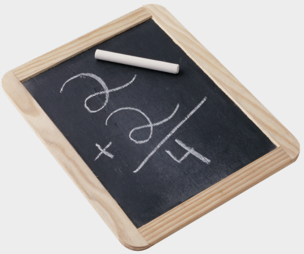
w^i

A

Width $\rho = \max_i \rho_i$

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

Details:



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 **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

→ 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

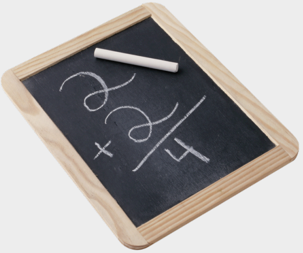
Crucial question: How large the worst-case overflow **ρ** can be?

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

In general: **ρ** 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: **ρ** is bounded by $\mathbf{O(n^{1/2}\, \varepsilon^{-1})}$ then

Proof: 

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

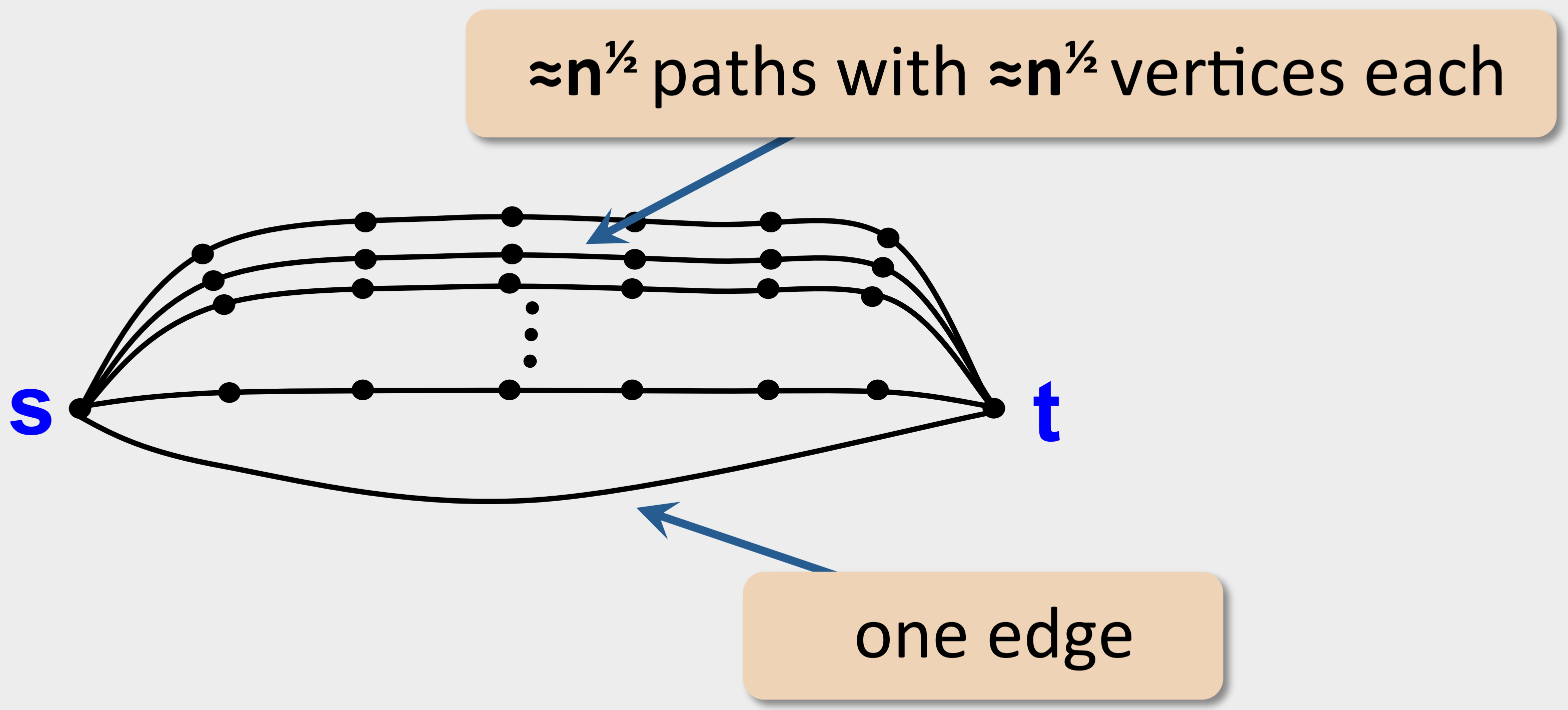
Going beyond the $\tilde{O}(n^{3/2})$ Barrier

Speeding up our algorithm

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

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

Not really…



Speeding up our algorithm

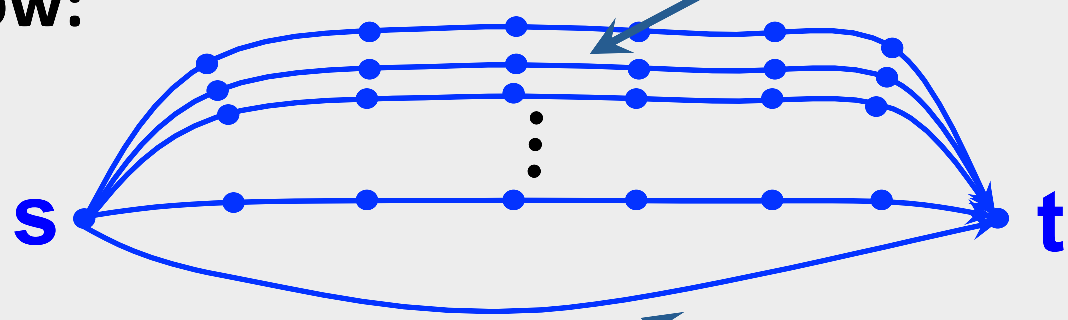
Running time is dominated by $\approx\rho$ elect. flow computations

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

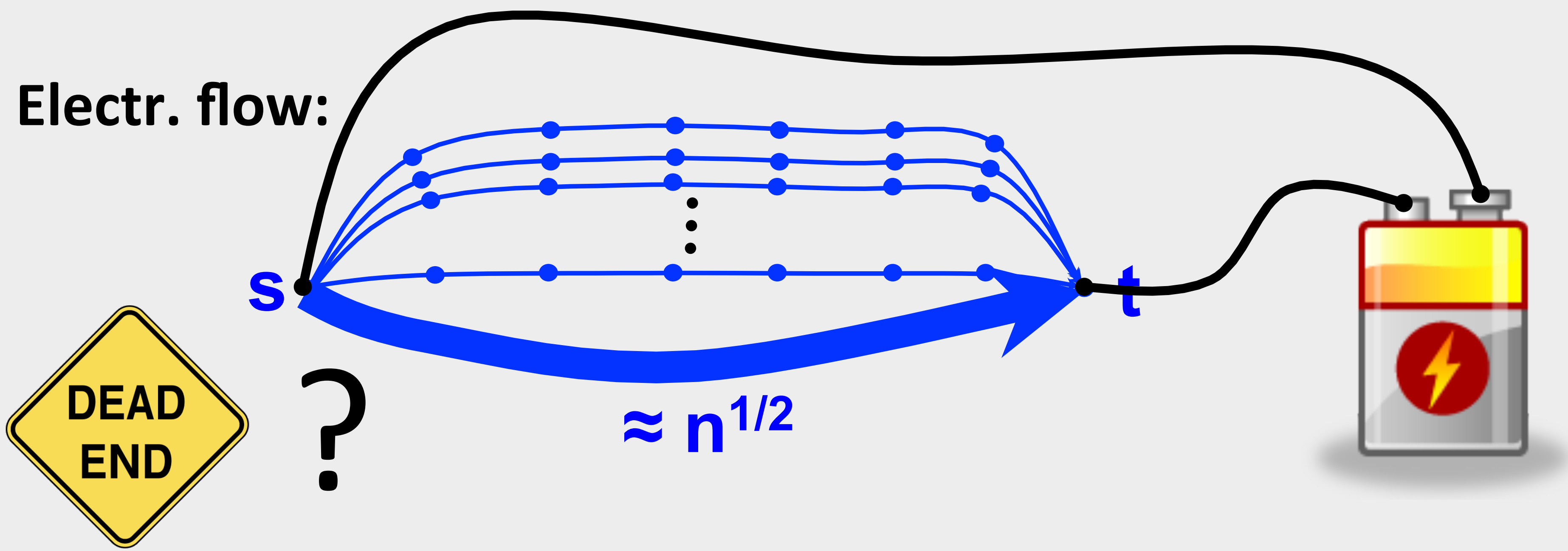
Speeding up our algorithm

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

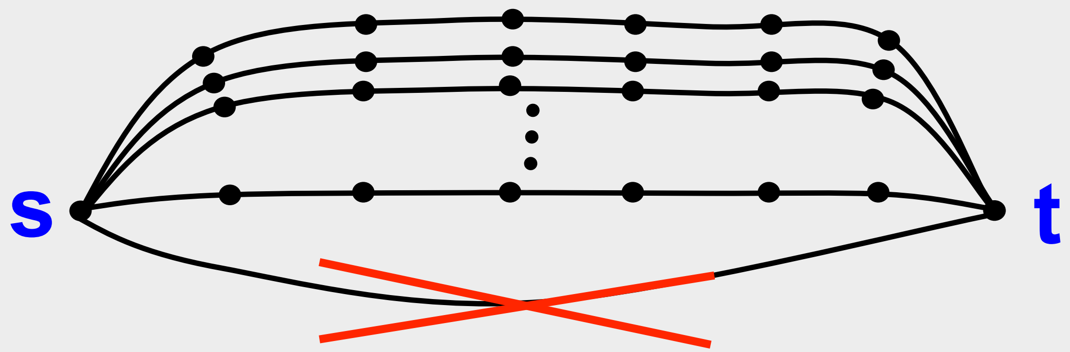
Can we improve our $\mathbf{O(n^{1/2}\,\varepsilon^{-1})}$ bound on $\boldsymbol{\rho}$?

Not really…

Electr. flow:



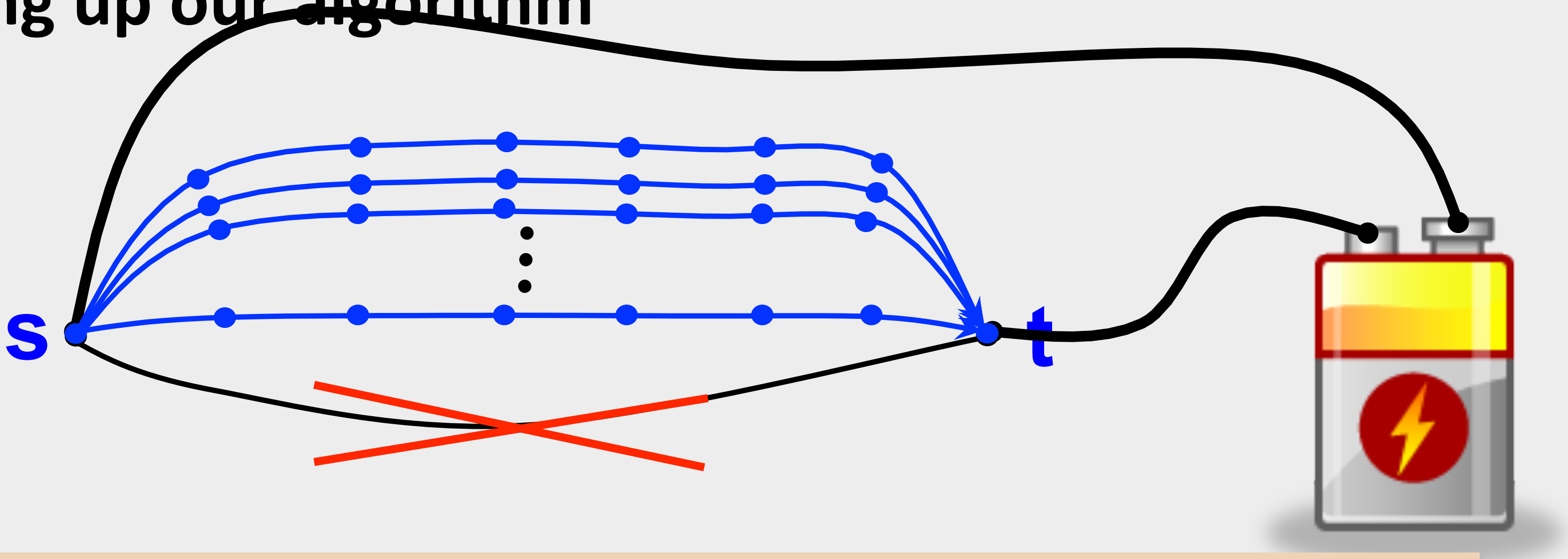
Speeding up our algorithm



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 **ρ' << ρ**

Modification of the oracle: If the computed electrical flow has some edge **e** flow more than **ρ'**:

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

→ **Recompute** the electrical flow

Note: If this oracle always **successfully** terminates, its effective overflow is **ρ'**

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

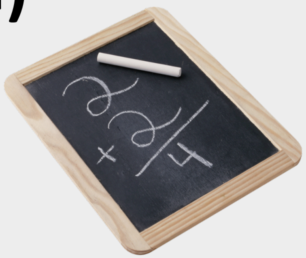
Need to show: Removal of an overflowing edge increases **E_r(f)** significantly

Speeding up our algorithm

Need to show: Removal of an overflowing edge increases $\mathbf{E_r(f)}$ significantly

Fact: If an edge **e** contributes a **δ-**fraction of energy then removing it increases $\mathbf{E_r(f)}$ by a factor of $\mathbf{1+\Omega(\delta)}$

Further: If an edge **e** flows at least **ρ'** in **f** then its energy contribution is $\mathbf{\Delta=\Omega(\varepsilon(\rho')^2/n)}$



Speeding up our algorithm

Need to show: Removal of an overflowing edge increases $\mathbf{E_r(f)}$ significantly

Fact: If an edge **e** contributes a **δ-**fraction of energy then removing it increases $\mathbf{E_r(f)}$ by a factor of $\mathbf{1+\Omega(\delta)}$

Further: If an edge **e** flows at least **ρ'** in **f** then its energy contribution is $\mathbf{\Delta=\Omega(\varepsilon(\rho')^2/n)}$

Putting it all together: We can have $\mathbf{\leq \tilde{O}(\Delta^{-1})=\tilde{O}(n/\varepsilon(\rho')^2)}$ edge removals before $\mathbf{E_r(f)}$ grows by too much

Taking $\mathbf{\rho' \approx n^{1/3}\varepsilon^{-1}}$ makes $\mathbf{\tilde{O}(\Delta^{-1})=\tilde{O}(\varepsilon n^{1/3})}$ be smaller than $\mathbf{\varepsilon F^* \leq \varepsilon\rho'}$ as needed

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

Thank you

Tomorrow: Computing an exact max flow