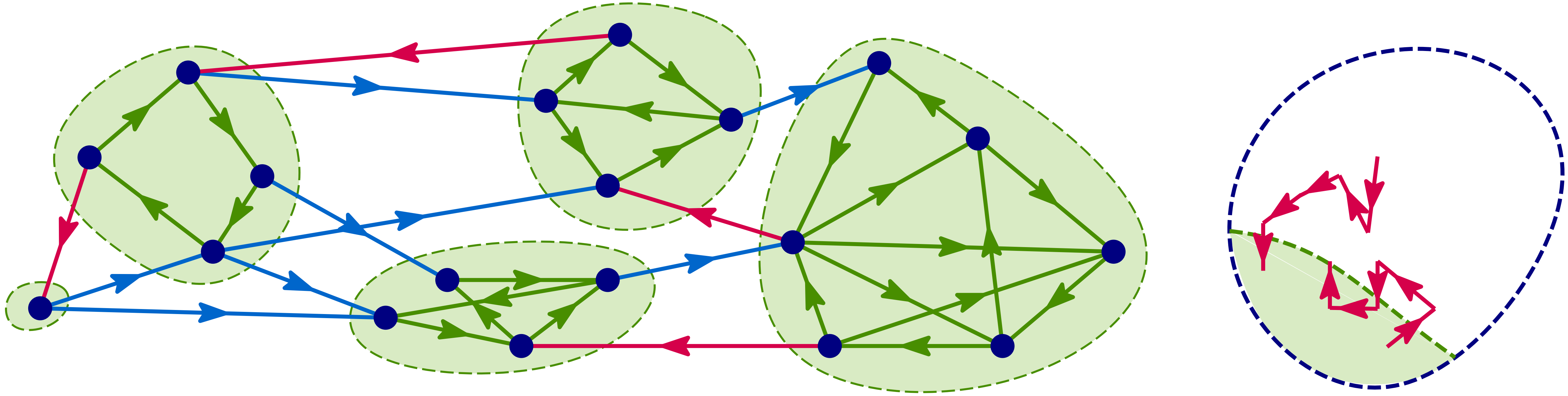
Combinatorial Maxflow in $n^{2+o(1)}$ Time

Aaron Bernstein
Joakim Blikstad[†]
Thatchaphol Saranurak
Ta-Wei Tu

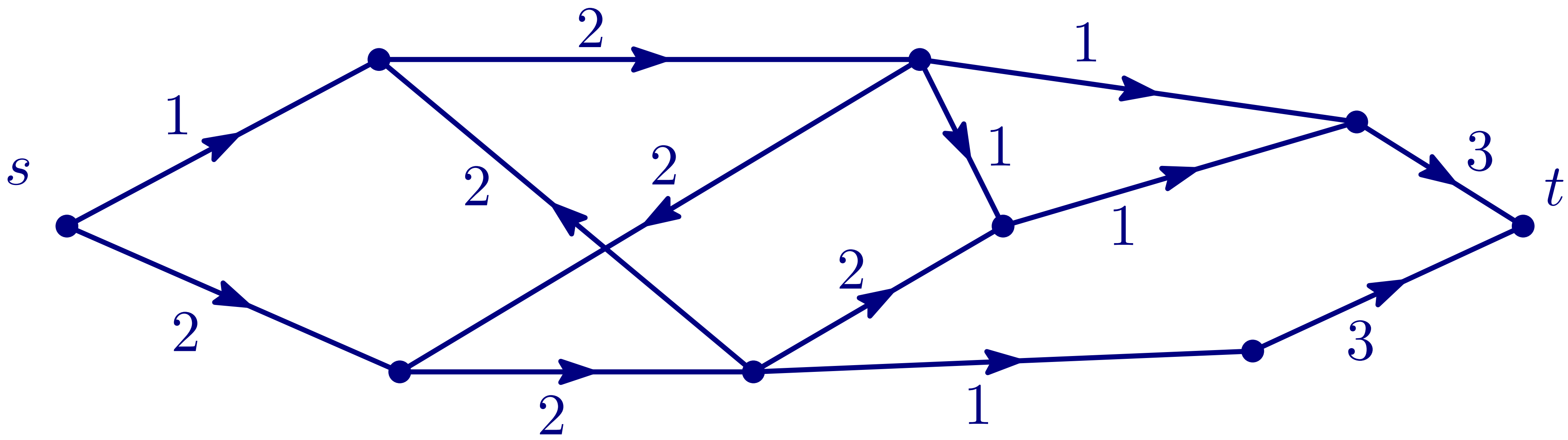


Algorithms & Complexity @ Warwick; Sep, 2024

[†]KTH Royal Institute of Technology

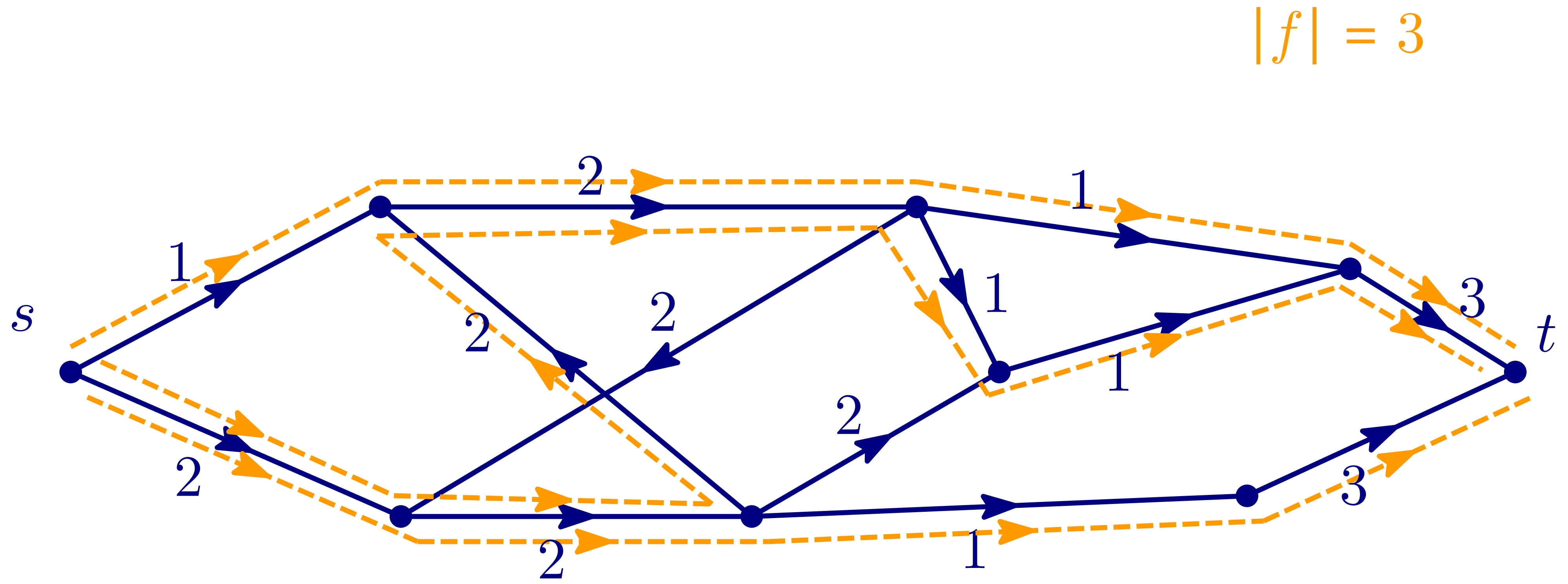
Maximum Flow

Given: Directed graph $G = (V, E)$, edge capacities $c : E \to \mathbb{Z}_{\geq 1}$, source s, and sink t.
Goal: Compute s, t-flow f of largest size.



Maximum Flow

Given: Directed graph $G = (V, E)$, edge capacities $c : E \to \mathbb{Z}_{\geq 1}$, source s, and sink t.
Goal: Compute s, t-flow f of largest size.



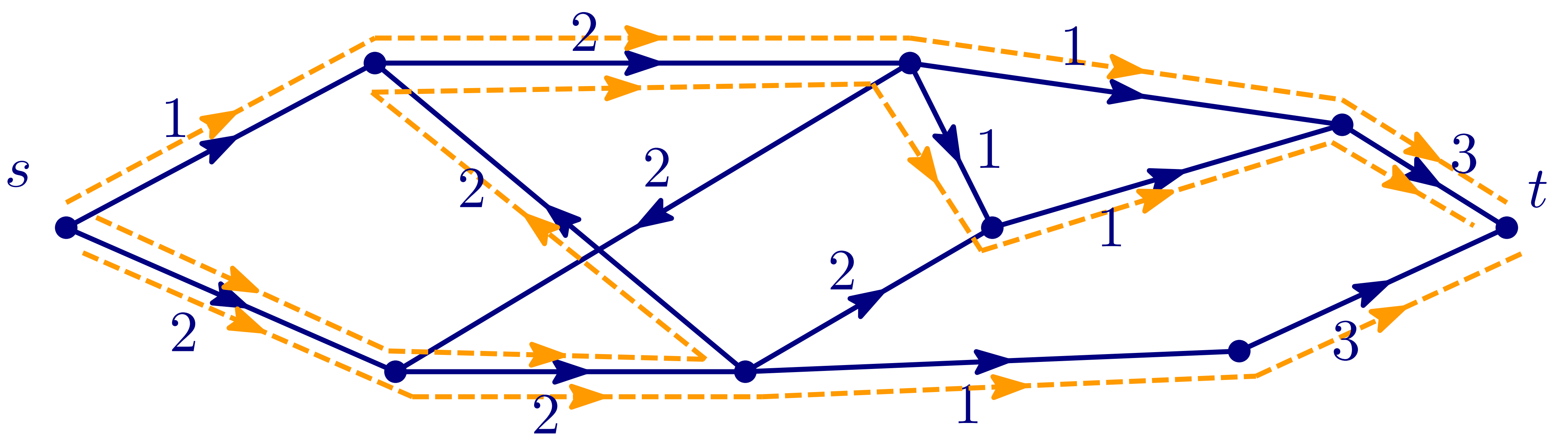
Maximum Flow

Given: Directed graph $G = (V, E)$, edge capacities $c : E \to \mathbb{Z}_{\geq 1}$, source s, and sink t.
Goal: Compute s, t-flow f of largest size.

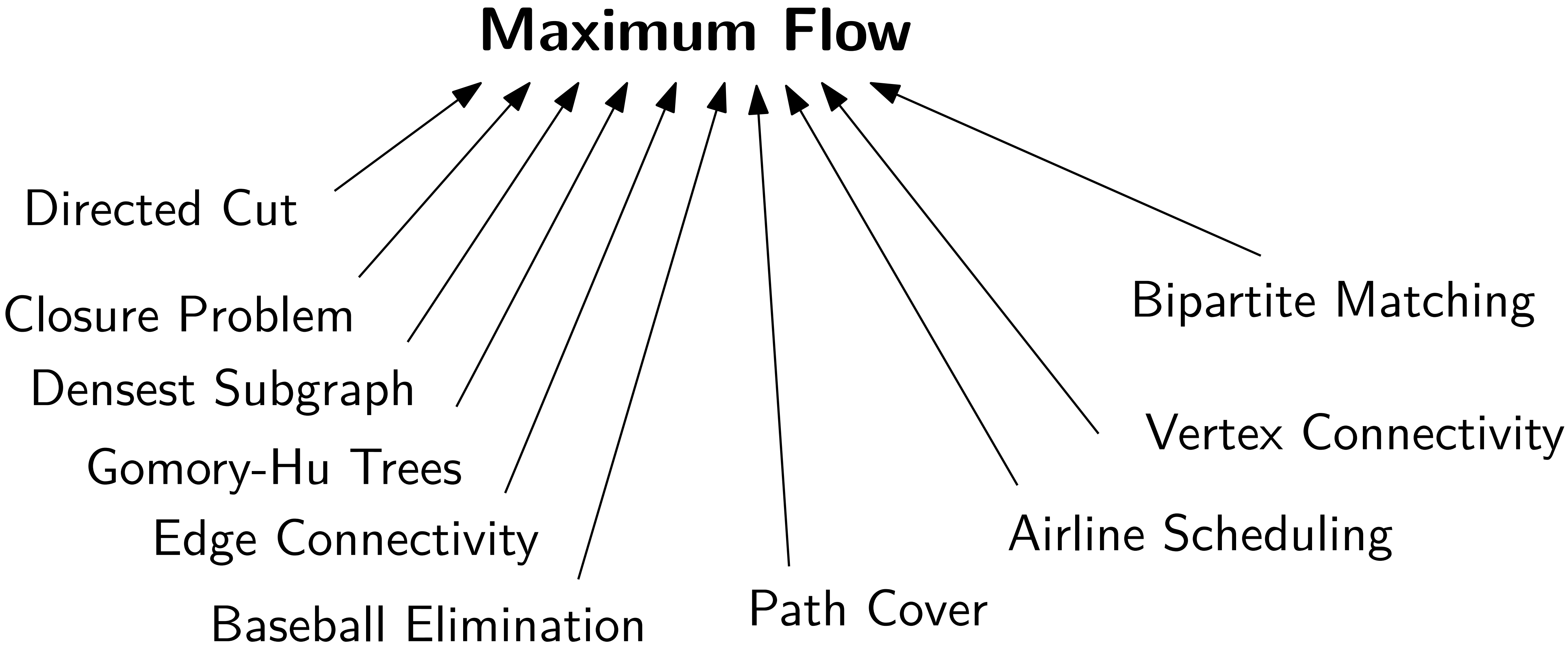
Flow satisfies:

(1) Capacity constraints $f(e) \leq c(e)$
(2) Conservation of flow "incoming $=$ outgoing"

$|f| = 3$



Reductions


Maximum Flow
Directed Cut
Closure Problem
Densest Subgraph
Gomory-Hu Trees
Edge Connectivity
Baseball Elimination
Path Cover
Airline Scheduling
Vertex Connectivity
Bipartite Matching

Maximum Flow — A Brief History

1955	Ford-Fulkerson	$O(E \cdot \|\text{answer}\|)$
1970	Edmonds-Karp	$O(VE^2)$
1970	Dinic "Blocking Flow"	$O(V^2E)$
1978	Malhotra-Kumar-Maheshwari	$O(V^3)$
1983	Dinics Dynamic Trees	$\tilde{O}(VE)$
1986	Goldberg-Tarjan "Push-Relabel"	$\tilde{O}(V^3)$
1988	Improved "Push-Relabel"	$\tilde{O}(VE)$
1998	Goldberg-Rao'	$\tilde{O}(E\sqrt{E})$ and $\tilde{O}(EV^{2/3})$

$\tilde{O}$: hides polylog

$\hat{O}$: hides $n^{o(1)}$

Maximum Flow — A Brief History

1955	Ford-Fulkerson	$O(E \cdot \|\text{answer}\|)$
1970	Edmonds-Karp	$O(VE^2)$
1970	Dinic "Blocking Flow"	$O(V^2E)$
1978	Malhotra-Kumar-Maheshwari	$O(V^3)$
1983	Dinics Dynamic Trees	$\tilde{O}(VE)$
1986	Goldberg-Tarjan "Push-Relabel"	$\tilde{O}(V^3)$
1988	Improved "Push-Relabel"	$\tilde{O}(VE)$
1998	Goldberg-Rao'	$\tilde{O}(E\sqrt{E})$ and $\tilde{O}(EV^{2/3})$
2014	Lee-Sidford	$\tilde{O}(E\sqrt{V})$
2020	Kathuria-Liu-Sidford	$\hat{O}(E^{4/3})$ (unit-capacity)
2020	BLNPSSSW / BLLSSSW	$\tilde{O}(E + V\sqrt{V})$
2022	Chen-Kyng-Liu-Peng-Gutenberg-Sachdeva	$\hat{O}(E)$

$\tilde{O}$: hides polylog

$\hat{O}$: hides $n^{o(1)}$

Maximum Flow — A Brief History

1955	Ford-Fulkerson	$O(E \cdot \lVert\text{answer}\rVert)$
1970	Edmonds-Karp	$O(VE^2)$
1970	Dinic "Blocking Flow"	$O(V^2E)$
1978	Malhotr	
1983	Dinics D	
1986	Goldber	
1988	Improved "Push-Relabel"	$\tilde{O}(VE)$
1998	Goldberg-Rao'	$\tilde{O}(E\sqrt{E})$ and $\tilde{O}(EV^{2/3})$
2014	Lee-Sidford	$\tilde{O}(E\sqrt{V})$
2020	Kathuria-Liu-Sidford	$\hat{O}(E^{4/3})$ (unit-capacity)
2020	BLNPSSSW / BLLSSSW	$\tilde{O}(E + V\sqrt{V})$
2022	Chen-Kyng-Liu-Peng-Gutenberg-Sachdeva	$\hat{O}(E)$

$\tilde{O}$: hides polylog

$\hat{O}$: hides $n^{o(1)}$

Combinatorial, Augmenting Paths

"Nice", "Simple", "Works well in practice"

Maximum Flow — A Brief History

1955	Ford-Fulkerson	$O(E \cdot \|\text{answer}\|)$
1970	Edmonds-Karp	$O(VE^2)$
1970	Dinic "Blocking Flow"	$O(V^2E)$
1978	Malhotr	
1983	Dinics D	
1986	Goldber	
1988	Improved "Push-Relabel"	$\tilde{O}(VE)$
1998	Goldberg-Rao'	$\tilde{O}(E\sqrt{E})$ and $\tilde{O}(EV^{2/3})$
2014	Lee-Sidf	$\tilde{O}(E\sqrt{V})$
2020	Kathuri	-capacity)
2020	BLNPS	
2022	Chen-Kyng-Liu-Peng-Gutenberg-Sachdeva	$O(E)$

$\tilde{O}$: hides polylog

$\hat{O}$: hides $n^{o(1)}$

Combinatorial, Augmenting Paths

"Nice", "Simple", "Works well in practice"

Continiuous Optimization, Interior Point Methods

"Precision issues", "I don't understand them"

Maximum Flow — A Brief History

$\tilde{O}$: hides polylog
$\hat{O}$: hides $n^{o(1)}$

1955	Ford-Fulkerson	$O(E \cdot \|\text{answer}\|)$
1970	Edmonds-Karp	$O(VE^2)$
1970	Dinic "Blocking Flow"	$O(V^2E)$
1978	Malhotr	
1983	Dinics D	
1986	Goldber	
1988	Improved "Push-Relabel"	$\tilde{O}(VE)$
1998	Goldberg-Rao'	$\tilde{O}(E\sqrt{E})$ and $\tilde{O}(EV^{2/3})$
2014	Lee-Sidf	
2020	Kathuri	-capacity)
2020	BLNPS	
2022	Chen-Ky	
2024	Bernstein-**B.**-Saranurak-Tu	$\hat{O}(V^2)$ "linear time in dense graphs"

Combinatorial, Augmenting Paths
"Nice", "Simple", "Works well in practice"

Continiuous Optimization, Interior Point Methods
"Precision issues", "I don't understand them"

New Era? — Comeback of Combinatorial Algorithms

2024 Bernstein-**B.**-Saranurak-Tu $\hat{O}(V^2)$ "linear time in dense graphs"

Main Result: Maximum flow in on n-vertex graphs in $n^{2+o(1)}$ time.

New Era? — Comeback of Combinatorial Algorithms

2024 Bernstein-**B.**-Saranurak-Tu $\hat{O}(V^2)$ "linear time in dense graphs"

Main Result: Maximum flow in on n-vertex graphs in $n^{2+o(1)}$ time.

Independent Work: $n^{2+o(1)}$ combinatorial bipartite matching [Chuzhoy-Khanna'24]

New Era? — Comeback of Combinatorial Algorithms

2024 Bernstein-**B.**-Saranurak-Tu $\hat{O}(V^2)$ "linear time in dense graphs"

Main Result: Maximum flow in on n-vertex graphs in $n^{2+o(1)}$ time.

Techniques:

Augmenting Paths (new version of Push-Relabel)

Directed Expander Hierarchy

Independent Work:
$n^{2+o(1)}$ combinatorial bipartite matching [Chuzhoy-Khanna'24]

New Era? — Comeback of Combinatorial Algorithms

2024 Bernstein-**B.**-Saranurak-Tu $\hat{O}(V^2)$ "linear time in dense graphs"

Main Result: Maximum flow in on n-vertex graphs in $n^{2+o(1)}$ time.

Techniques:

Augmenting Paths (new version of Push-Relabel)

Directed Expander Hierarchy

My Hope: (in a few years)

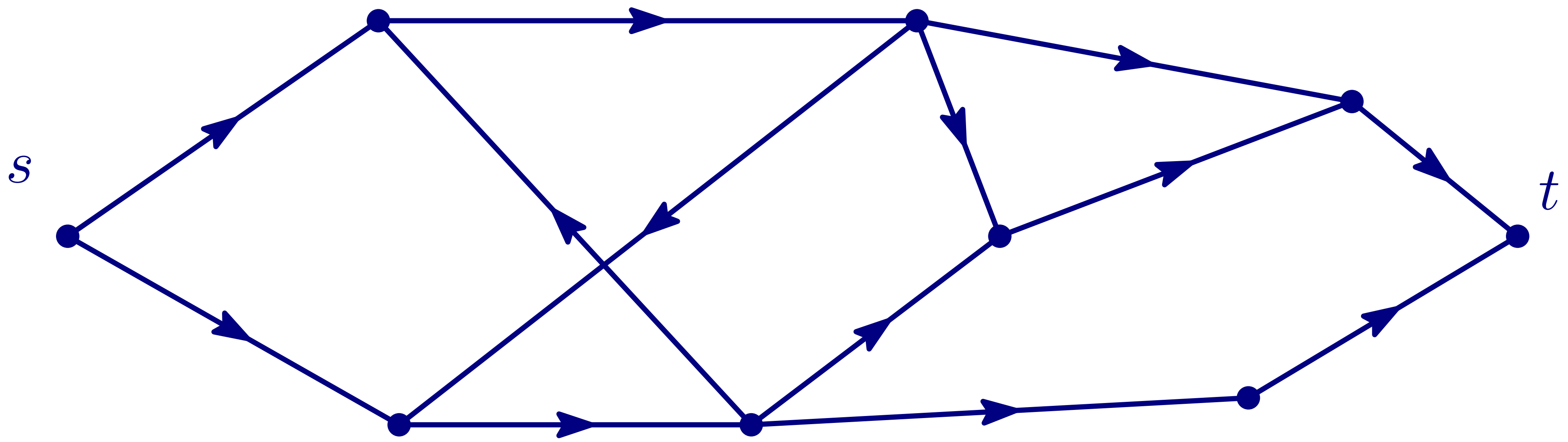
"Simple", "Combinatorial" $\tilde{O}(E)$ Maximum Flow?

Non-bipartite Maximum Matching in $\tilde{O}(V^2)$ or $\tilde{O}(E)$ time?

Independent Work:
$n^{2+o(1)}$ combinatorial bipartite matching [Chuzhoy-Khanna'24]

Augmenting Paths

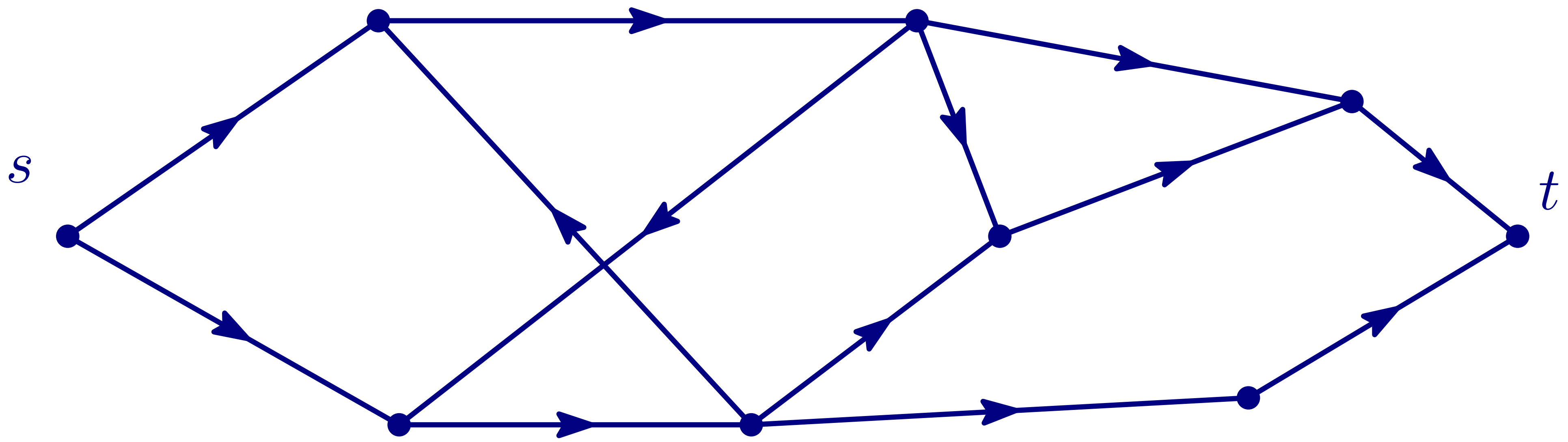
[Ford-Fulkerson 1955]
[Jacobi 1836]



Augmenting Paths

[Ford-Fulkerson 1955]
[Jacobi 1836]

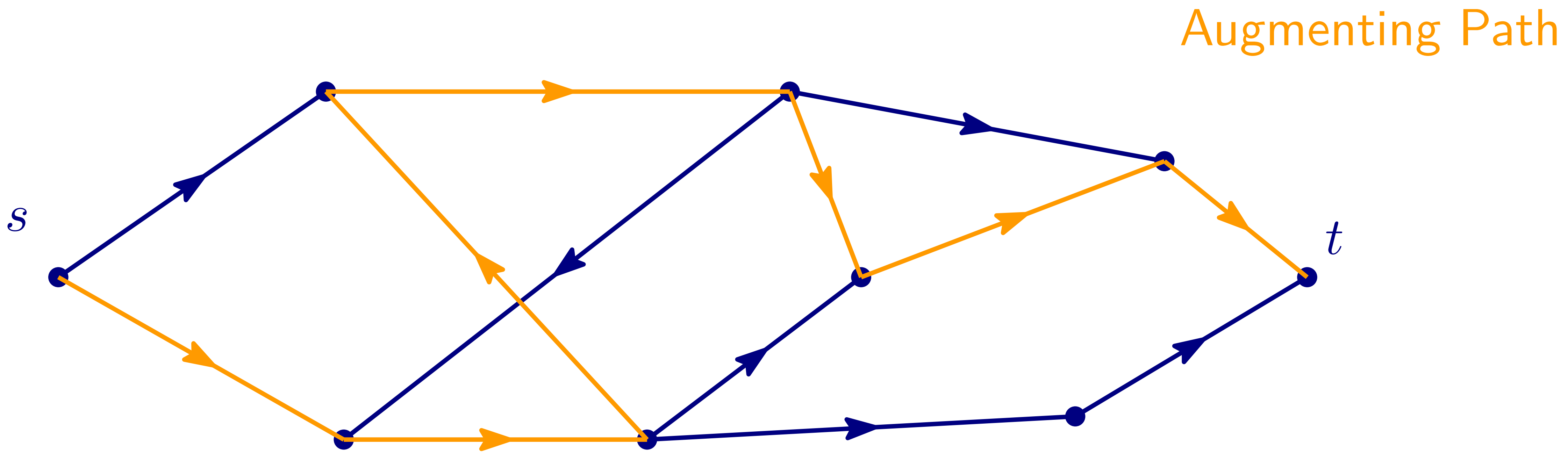
Remainder of this talk: unit-capacities $c(e) = 1$



Augmenting Paths

[Ford-Fulkerson 1955]
[Jacobi 1836]

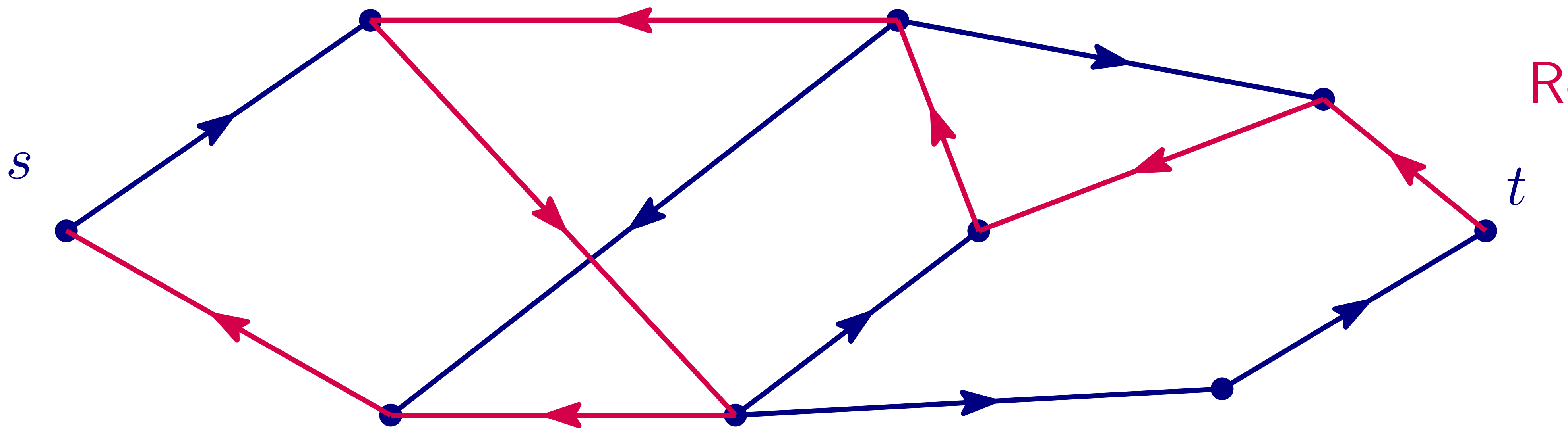
Remainder of this talk: unit-capacities $c(e) = 1$



Augmenting Paths

[Ford-Fulkerson 1955]
[Jacobi 1836]

Remainder of this talk: unit-capacities $c(e) = 1$



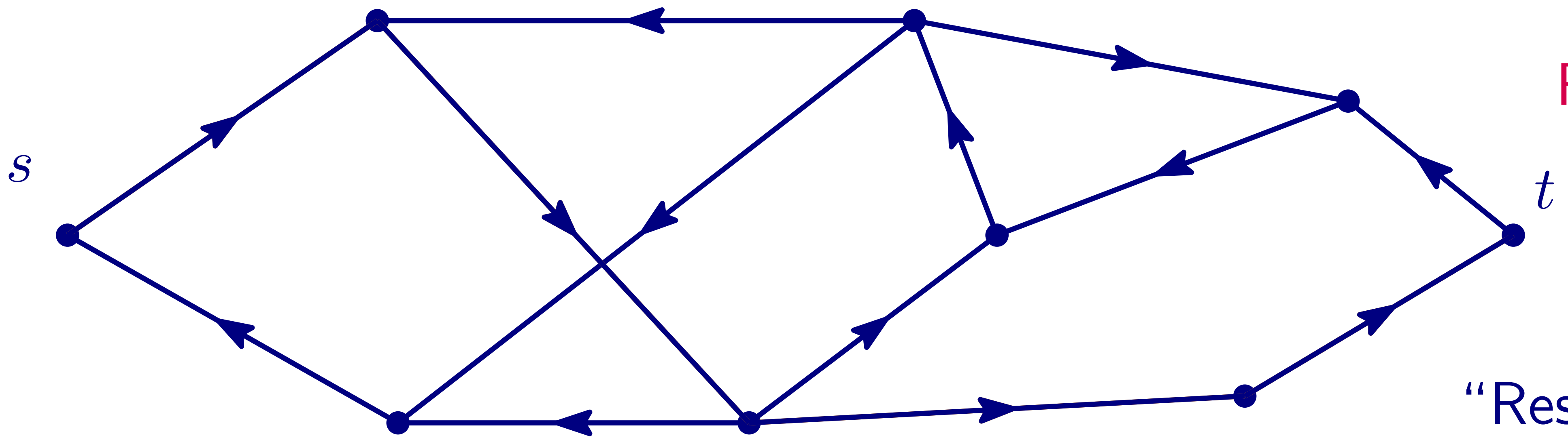
Augmenting Paths

[Ford-Fulkerson 1955]
[Jacobi 1836]

Remainder of this talk: unit-capacities $c(e) = 1$

Augmenting Path

Reverse it

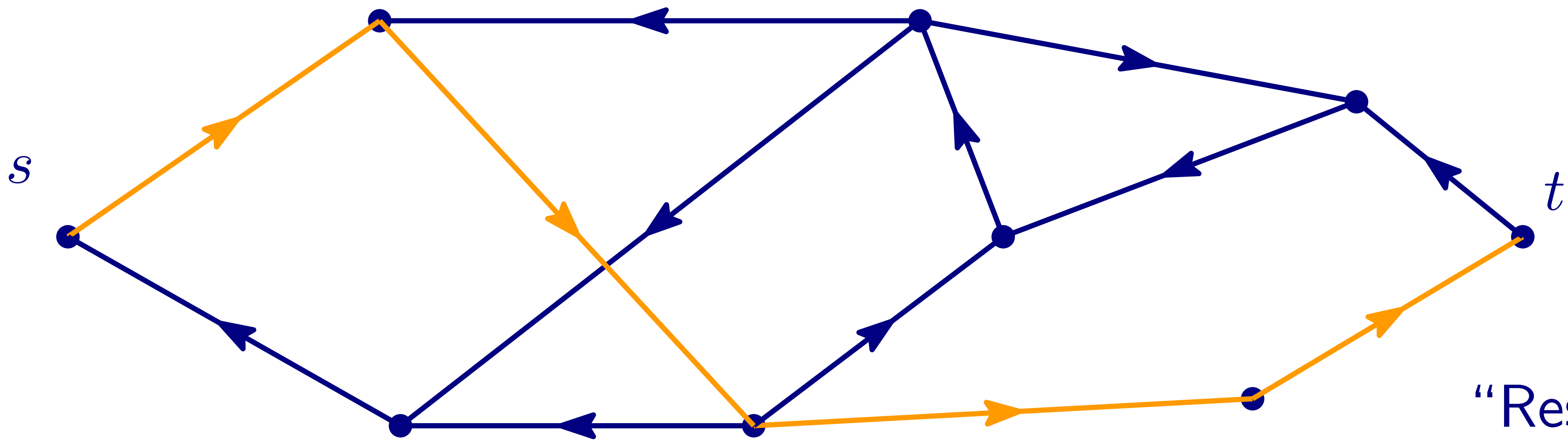


"Residual Graph"

Augmenting Paths

[Ford-Fulkerson 1955]
[Jacobi 1836]

Remainder of this talk: unit-capacities $c(e) = 1$



Augmenting Path

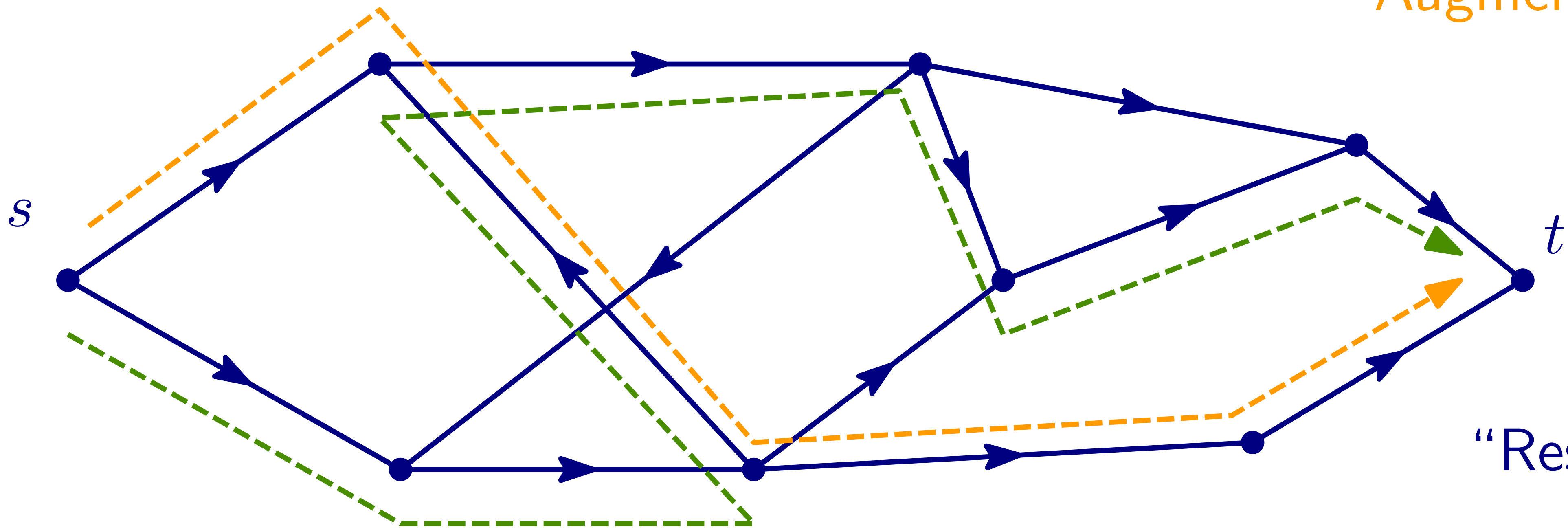
Reverse it

"Residual Graph"

Augmenting Paths

[Ford-Fulkerson 1955]
[Jacobi 1836]

Remainder of this talk: unit-capacities $c(e) = 1$



Augmenting Path

Reverse it

"Residual Graph"

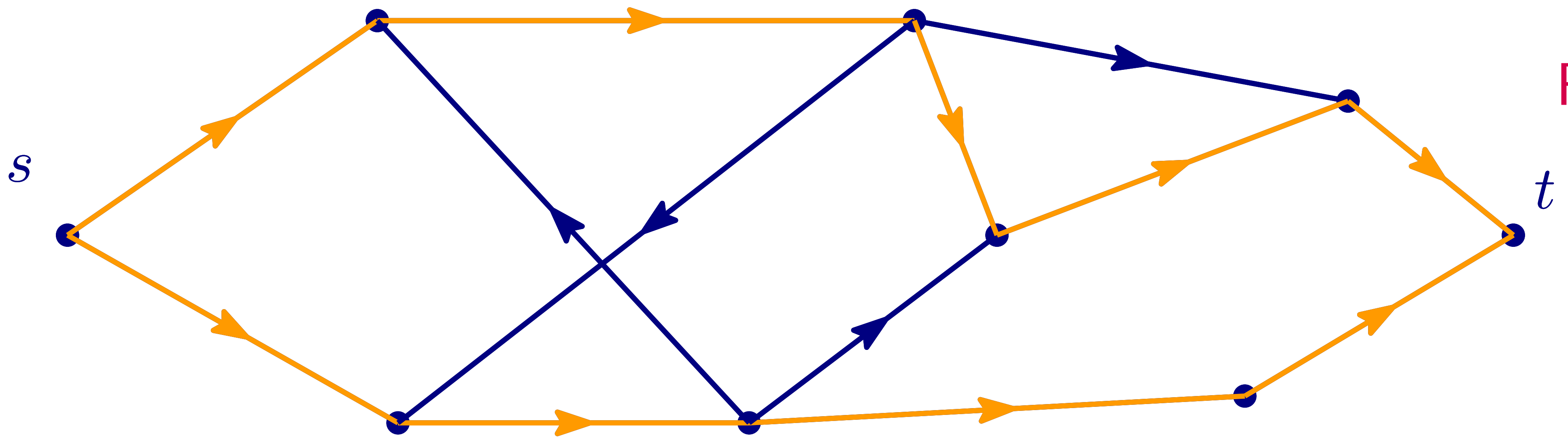
Augmenting Paths

[Ford-Fulkerson 1955]
[Jacobi 1836]

Remainder of this talk: unit-capacities $c(e) = 1$

Augmenting Path

Reverse it



Maximum Flow f, $|f| = 2$

Approximate Flow $\Longrightarrow$ Exact Flow

Approximate Flow $\implies$ Exact Flow

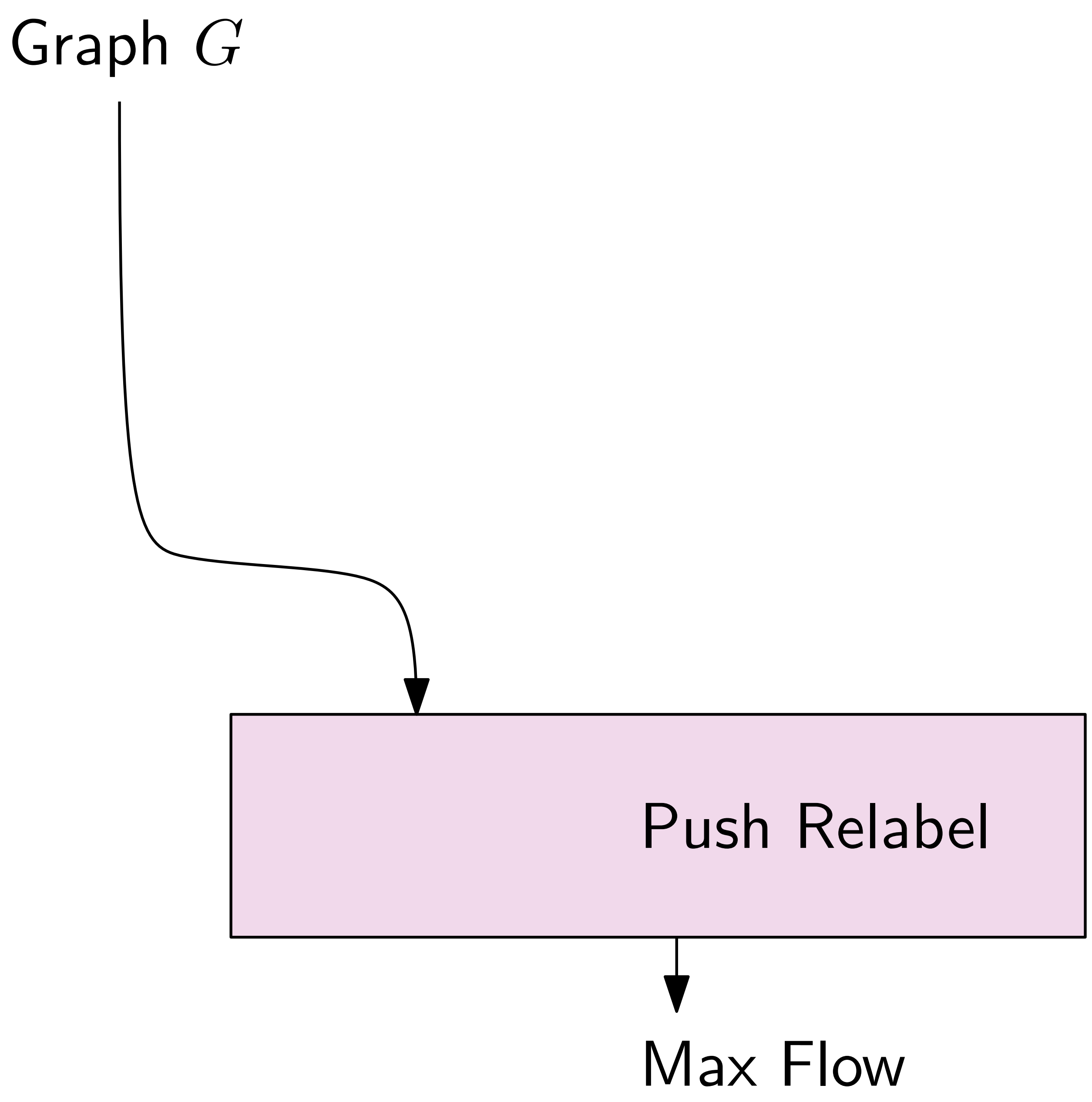
Proof. Recurse on residual graph.

Goal in rest of talk: constant- or $\frac{1}{n^{o(1)}}$-approx flow.

(does not work in undirected graphs)

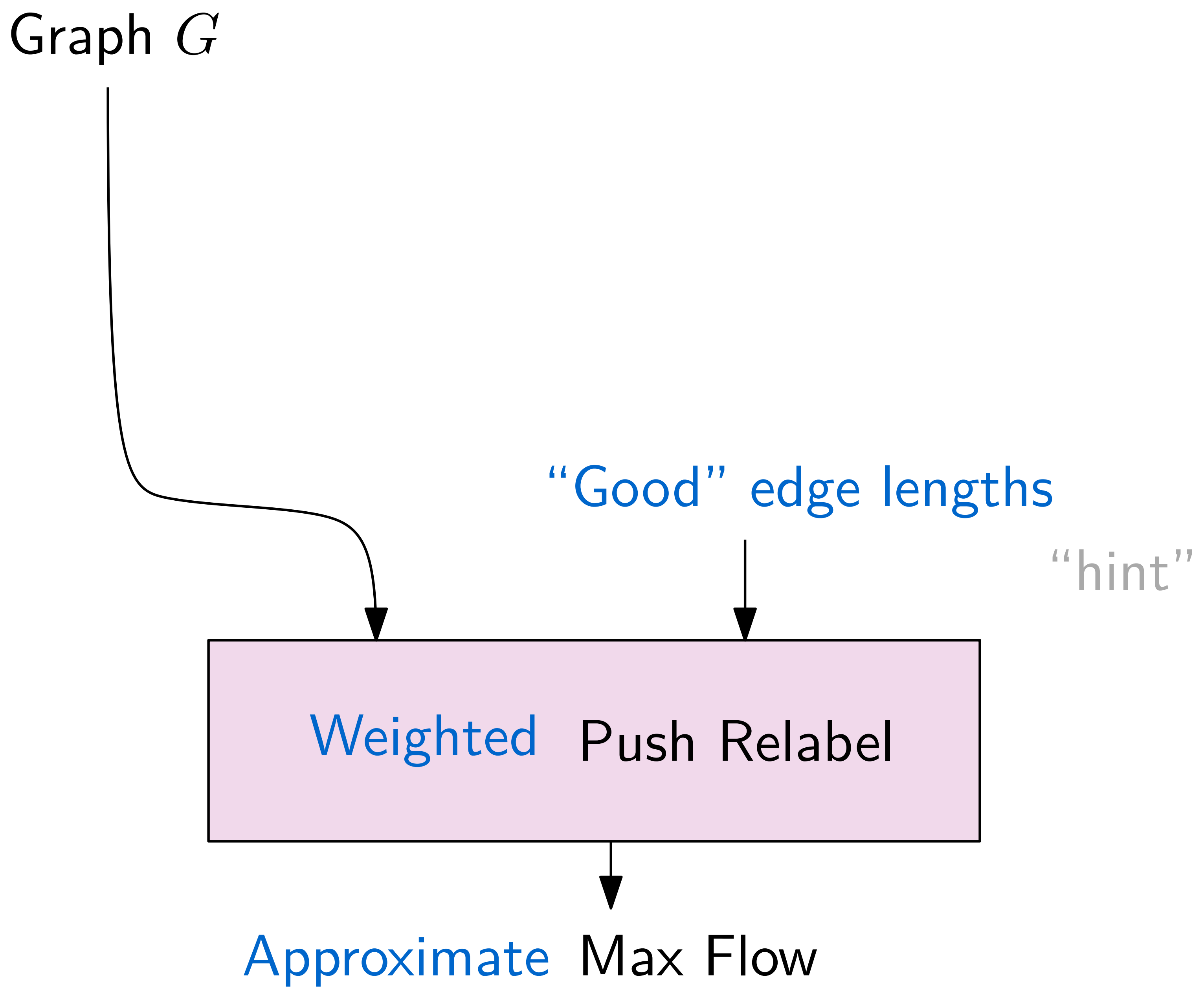
Outline

1. Recap: Push-Relabel



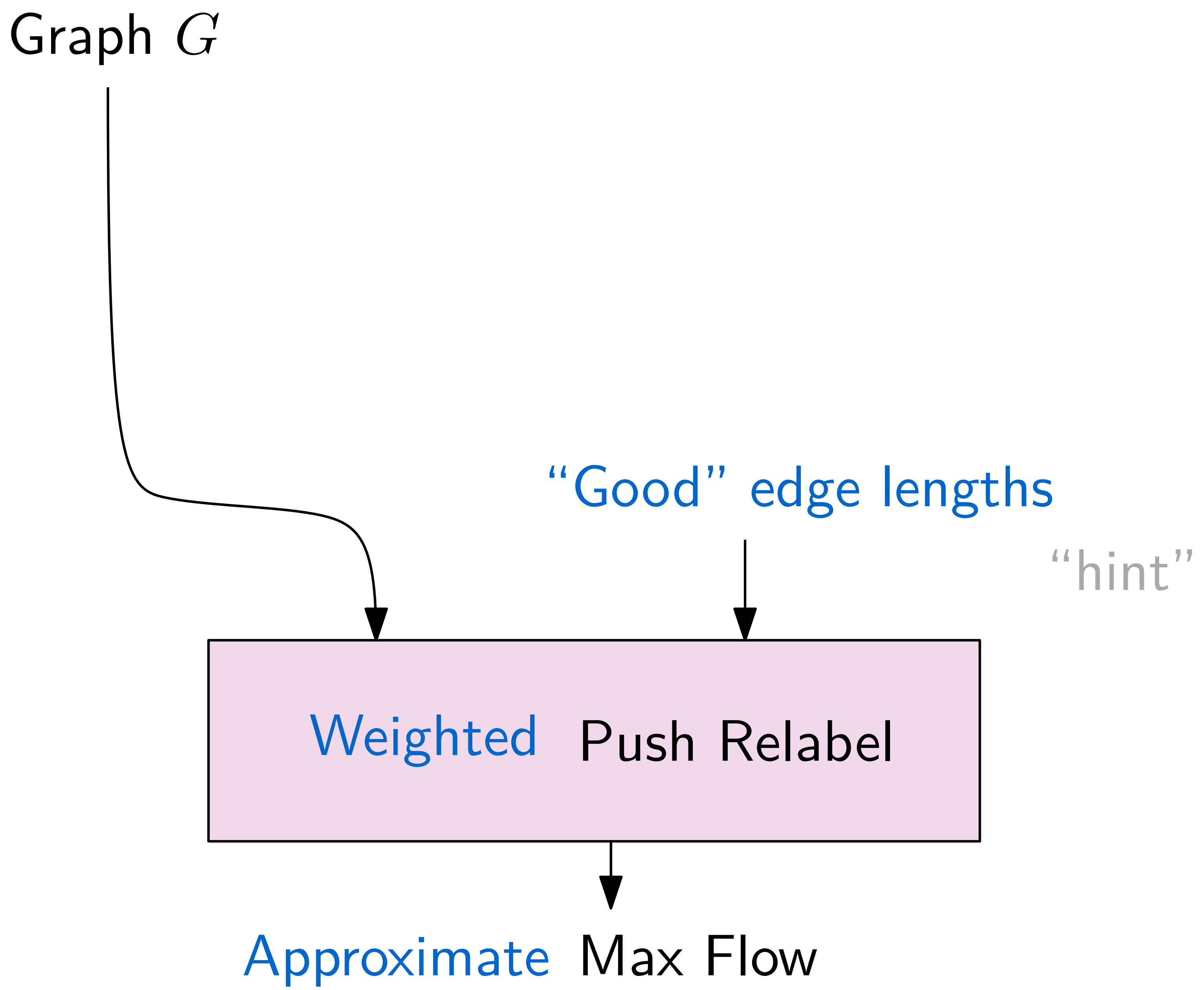
Outline

1. Recap: Push-Relabel
2. Weighted PR



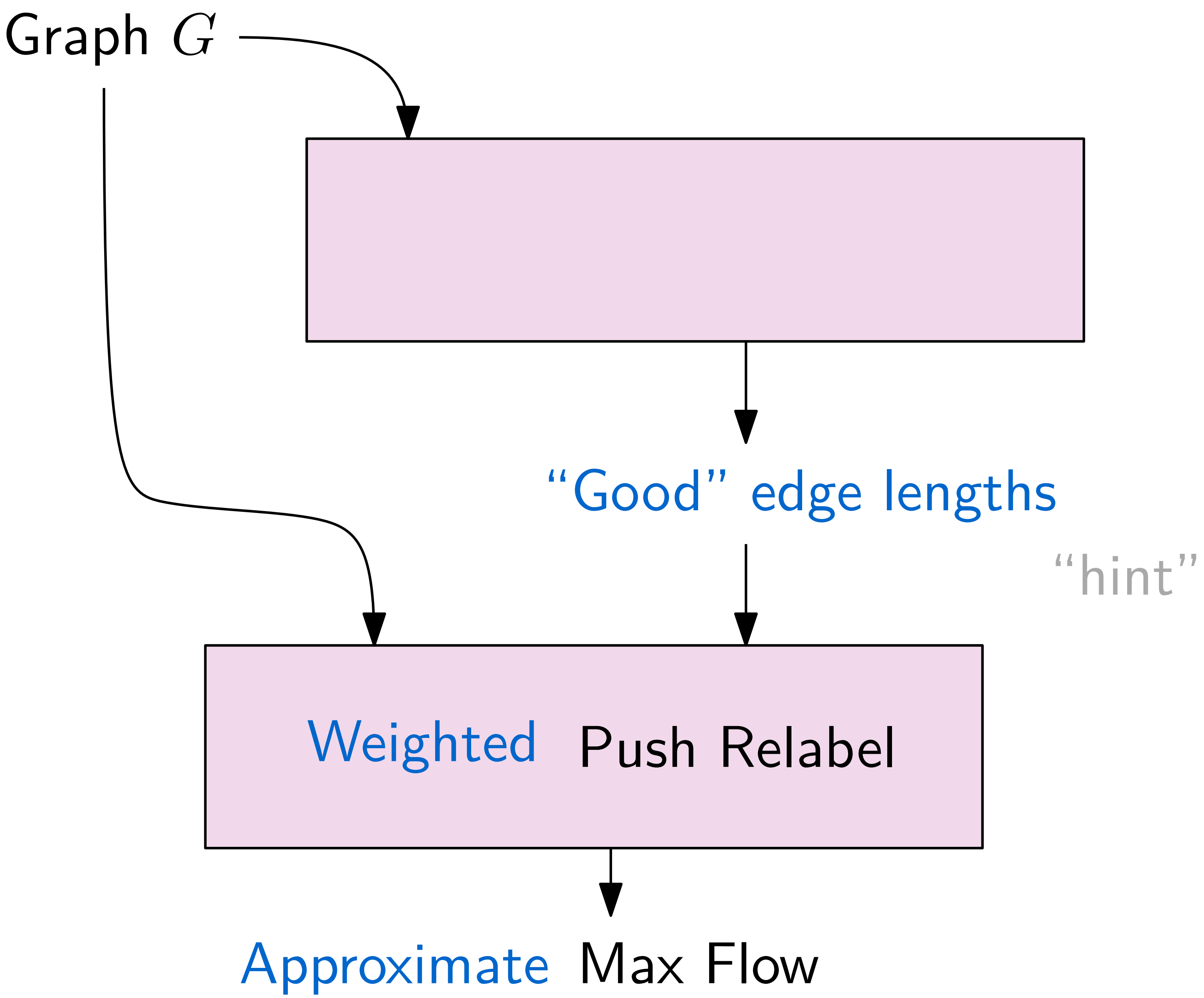
Outline

1. Recap: Push-Relabel
2. Weighted PR
3. "Good"



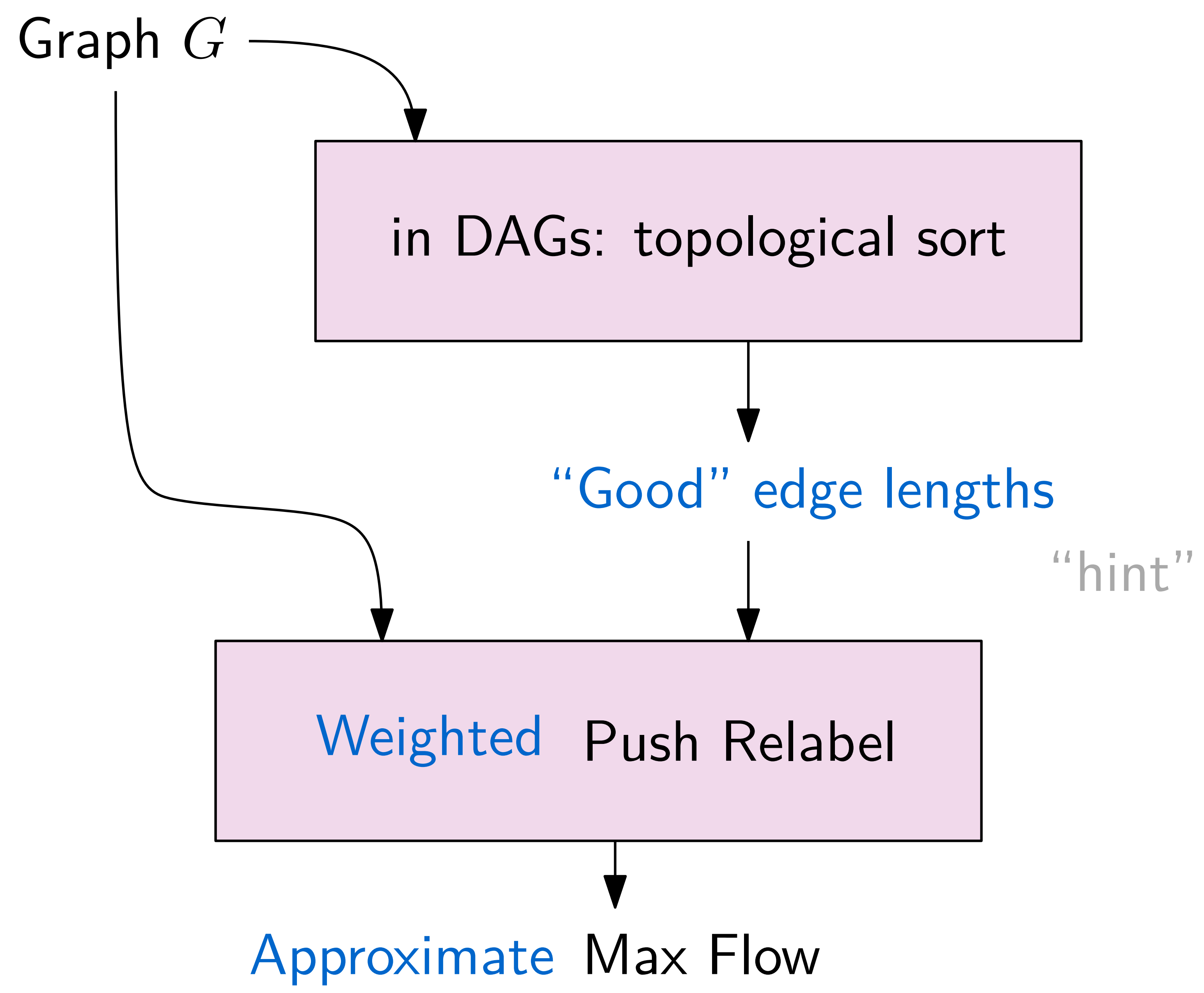
Outline

1. Recap: Push-Relabel
2. Weighted PR
3. “Good”



Outline

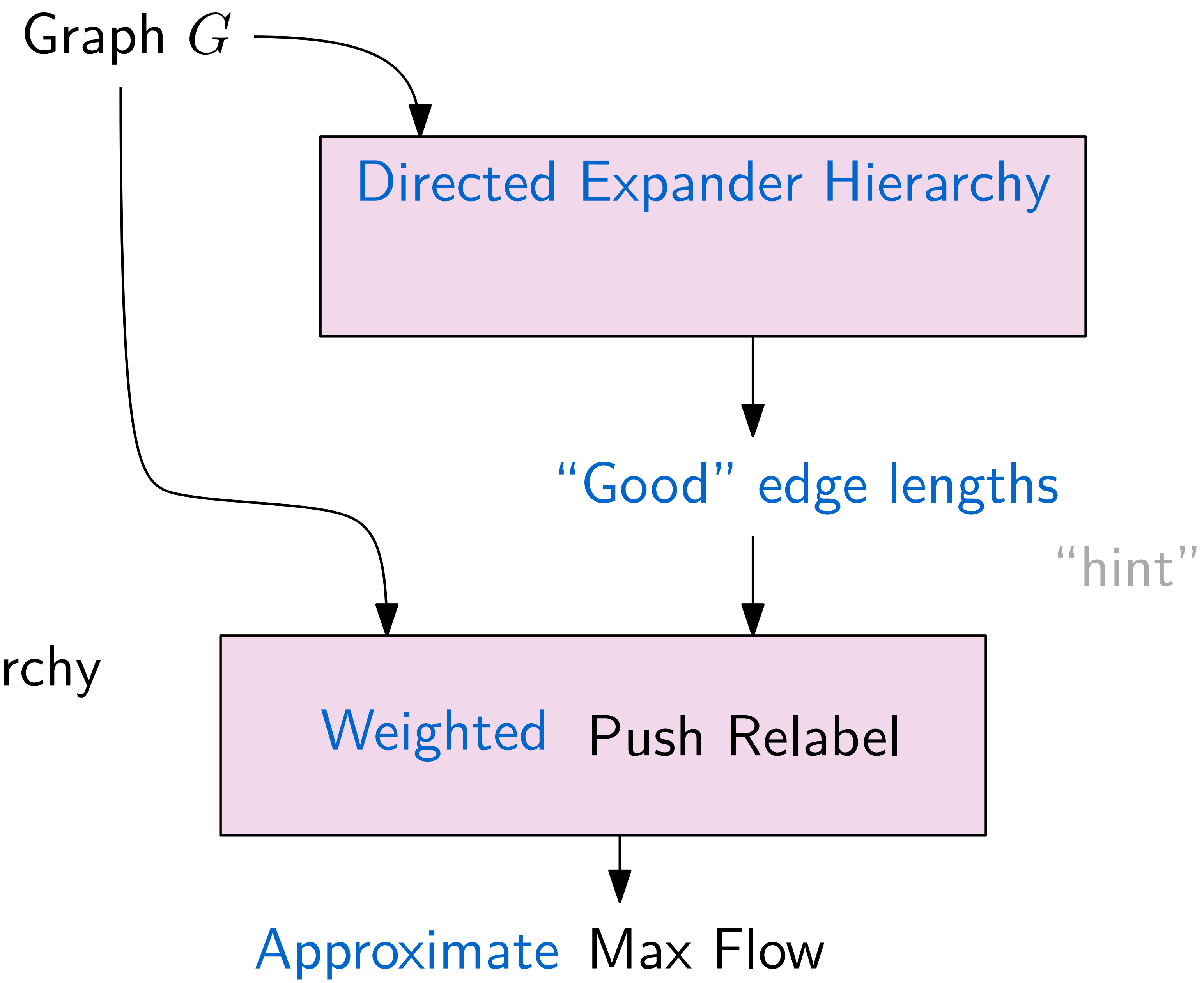
1. Recap: Push-Relabel
2. Weighted PR
3. “Good”
4. Edge Lenghts in DAGs



Outline

1. Recap: Push-Relabel
2. Weighted PR
3. "Good"
4. Edge Lenghts in DAGs
5. General Graphs:

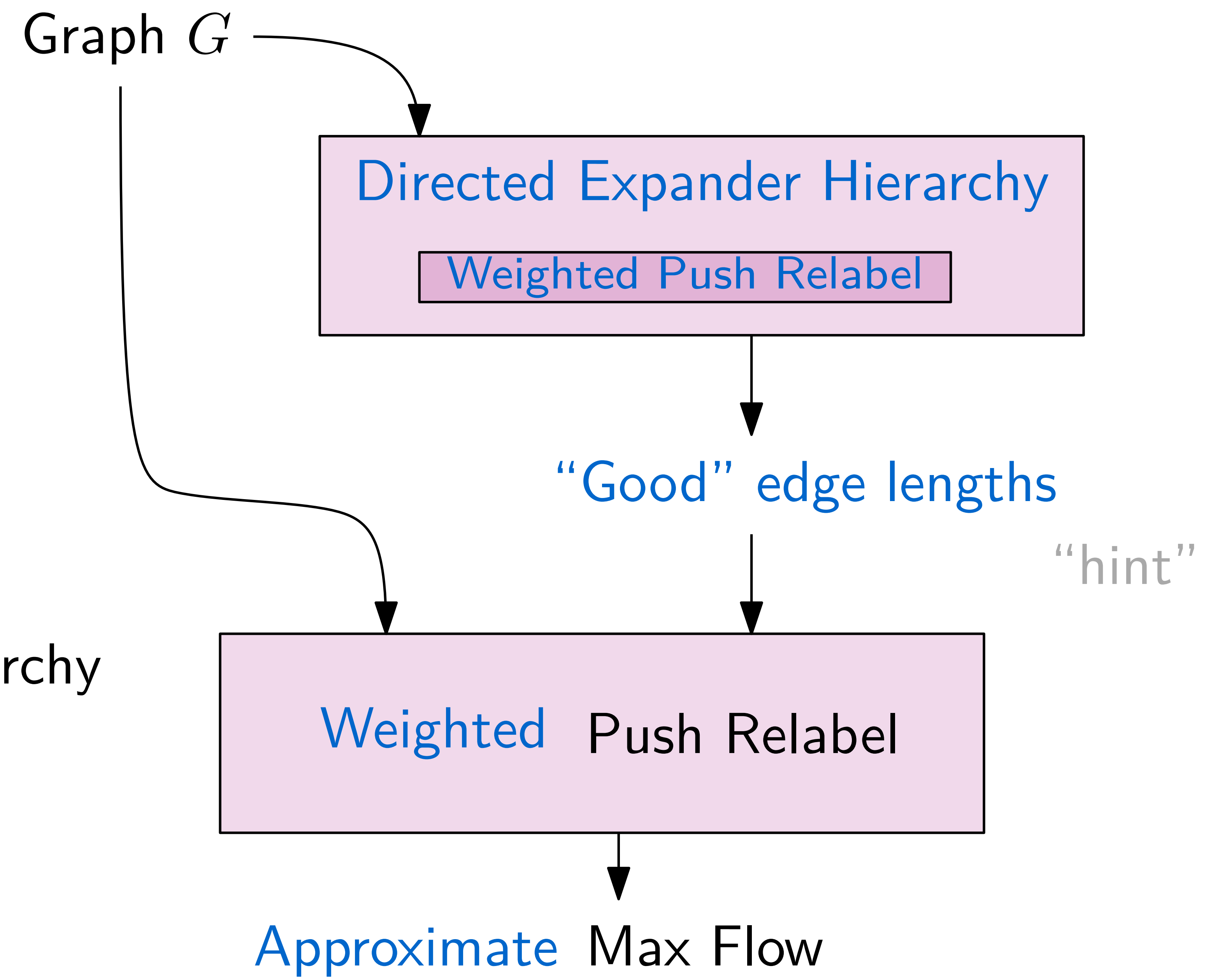
   Directed Expander Hierarchy



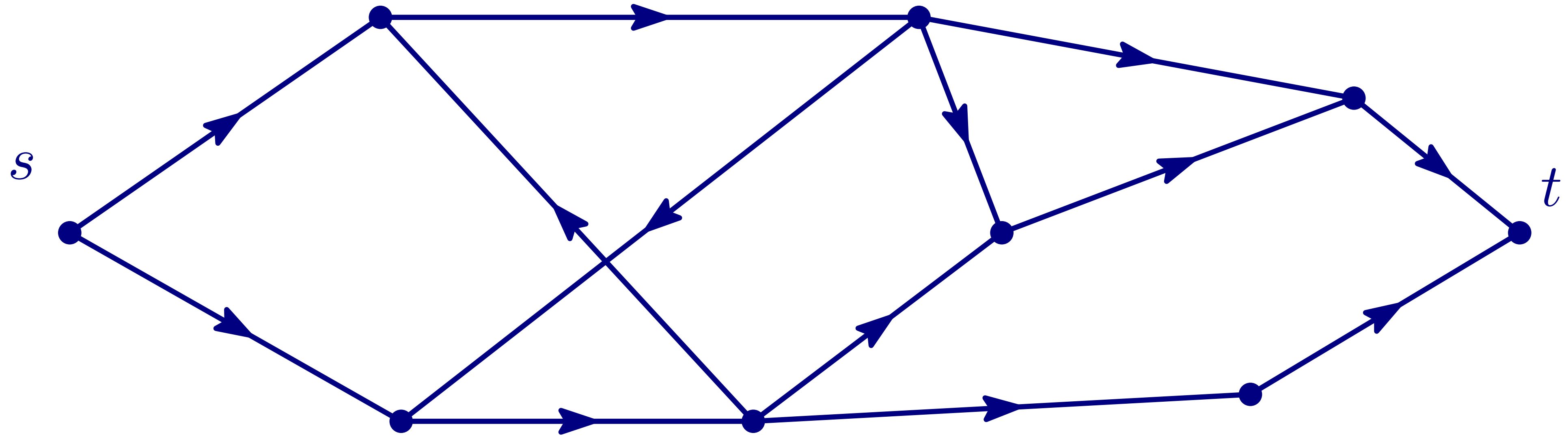
Outline

1. Recap: Push-Relabel
2. Weighted PR
3. "Good"
4. Edge Lenghts in DAGs
5. General Graphs:

    Directed Expander Hierarchy

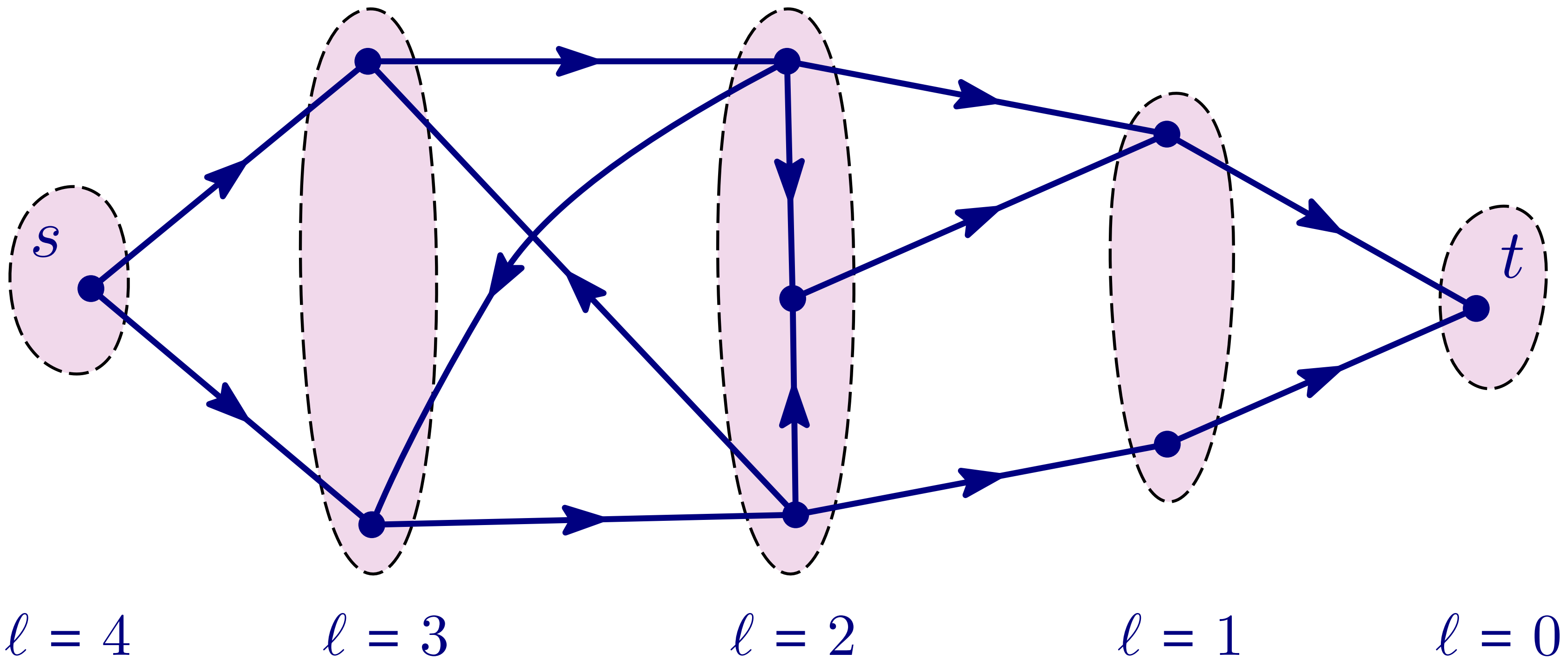


Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88]



Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88]

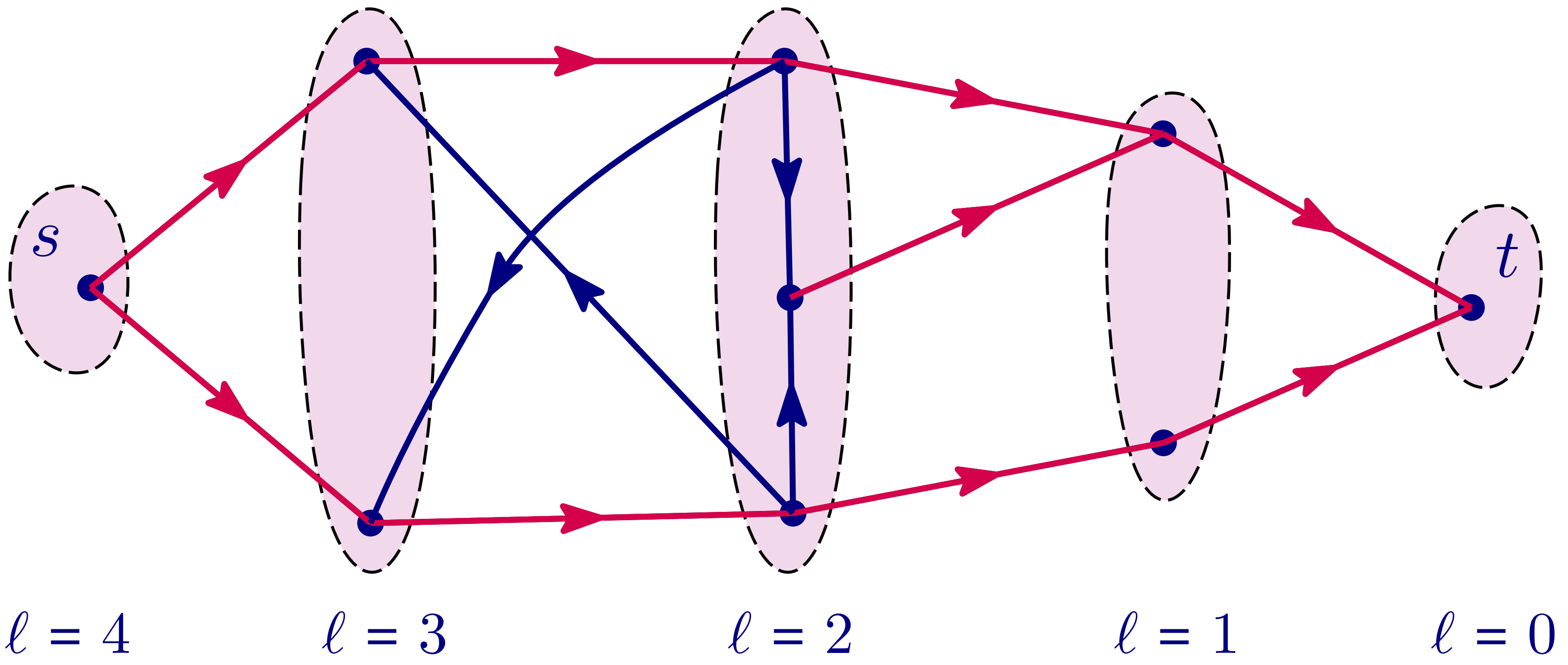
$\ell(v) = \text{dist}(v, t)$



Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88]

edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + 1$

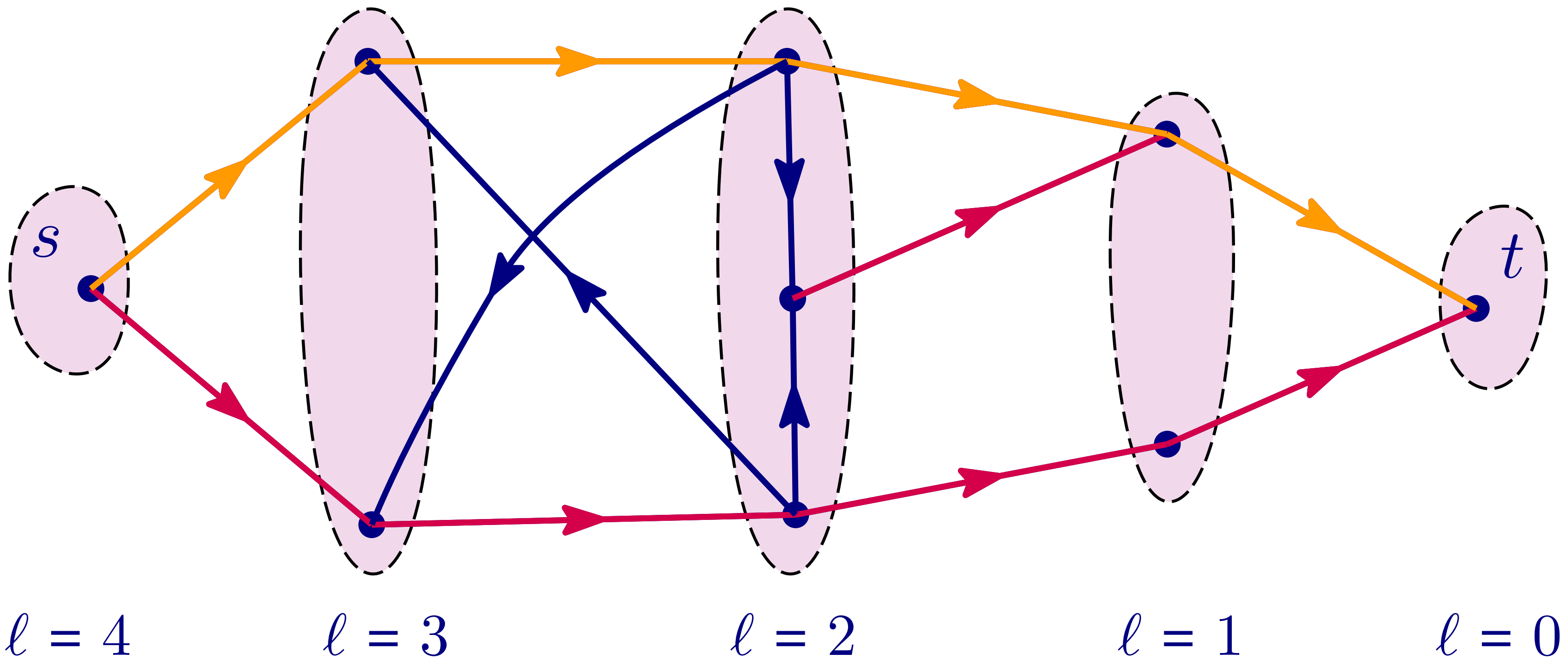
$\ell(v) = \text{dist}(v, t)$



Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88]

edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + 1$

$\ell(v) = \mathrm{dist}(v, t)$



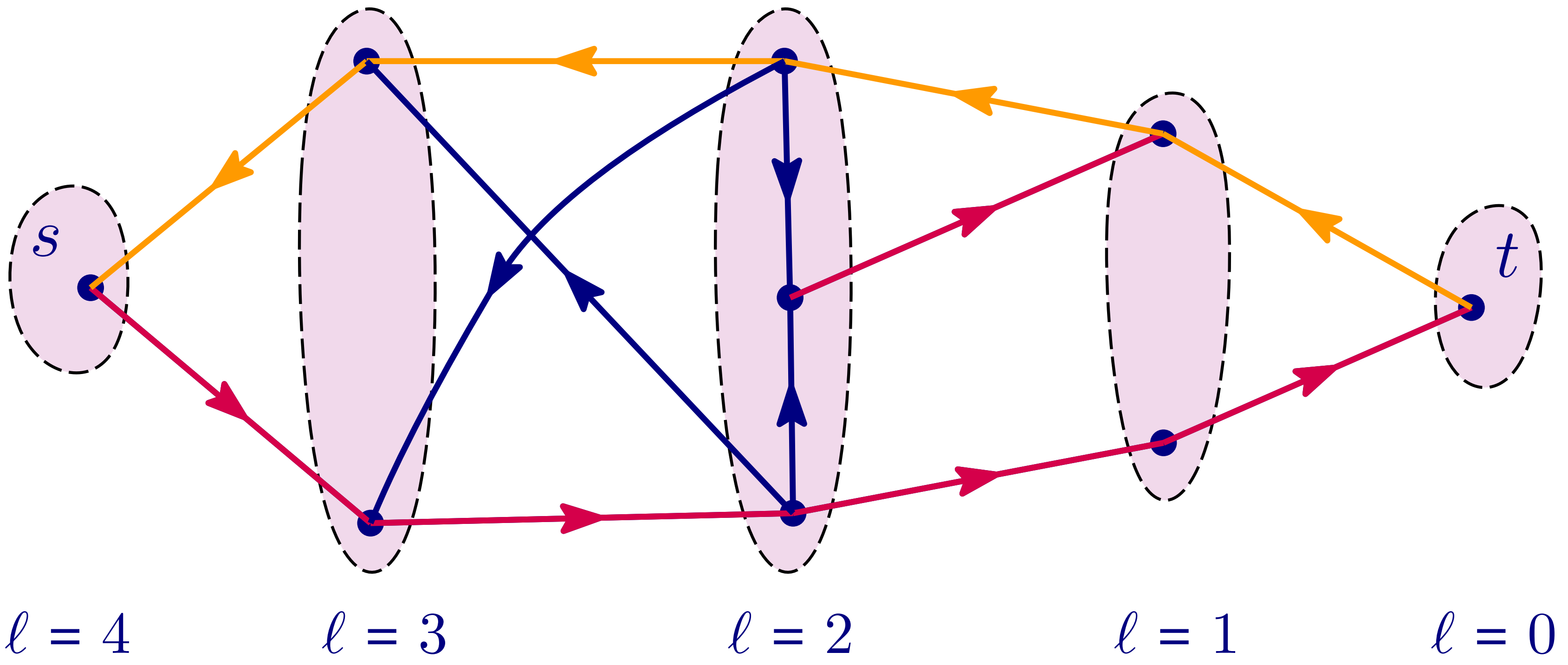
Shortest Augmenting Path: follow admissible edges from s

Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88]

edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + 1$

$\ell(v) = \mathrm{dist}(v, t)$

Reverse it



Shortest Augmenting Path: follow admissible edges from s

Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88]

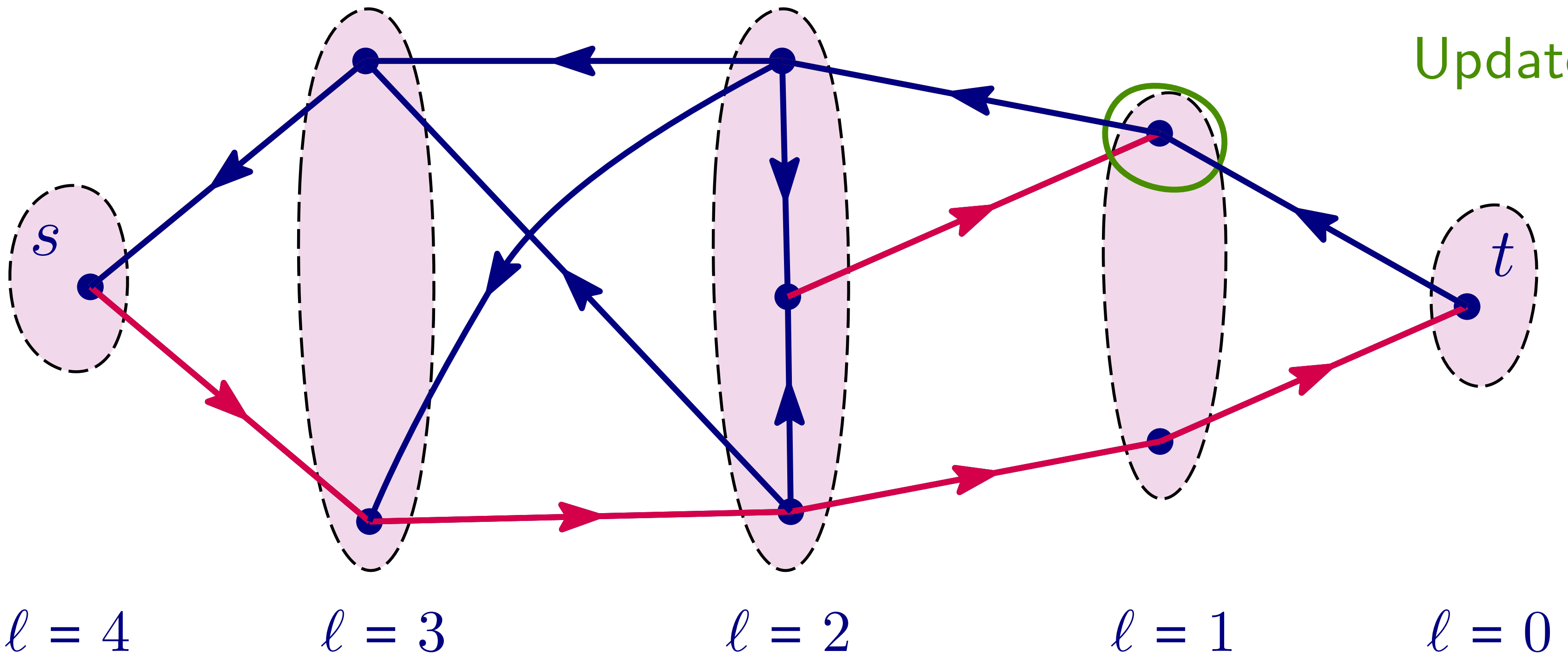
edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + 1$

$\ell(v) = \mathrm{dist}(v, t)$

Reverse it

Update ℓ?

Only increases!



Shortest Augmenting Path: follow admissible edges from s

Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88]

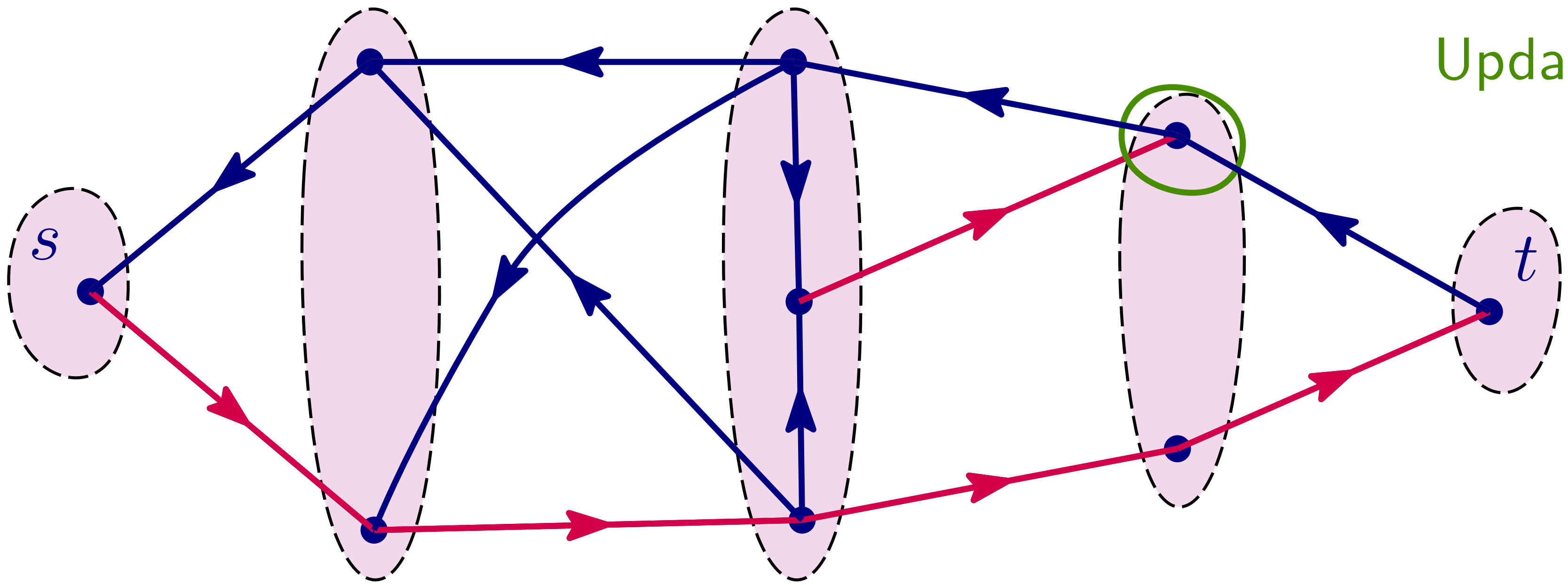
$\ell(v) = \text{dist}(v, t)$

edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + 1$

Reverse it

Update ℓ?

Only increases!



$\ell = 4$ $\ell = 3$ $\ell = 2$ $\ell = 1$ $\ell = 0$

Shortest Augmenting Path: follow admissible edges from s

RELABEL(v)
If no admissible out-edge:
$\ell(v) \leftarrow \ell(v) + 1$

Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88]

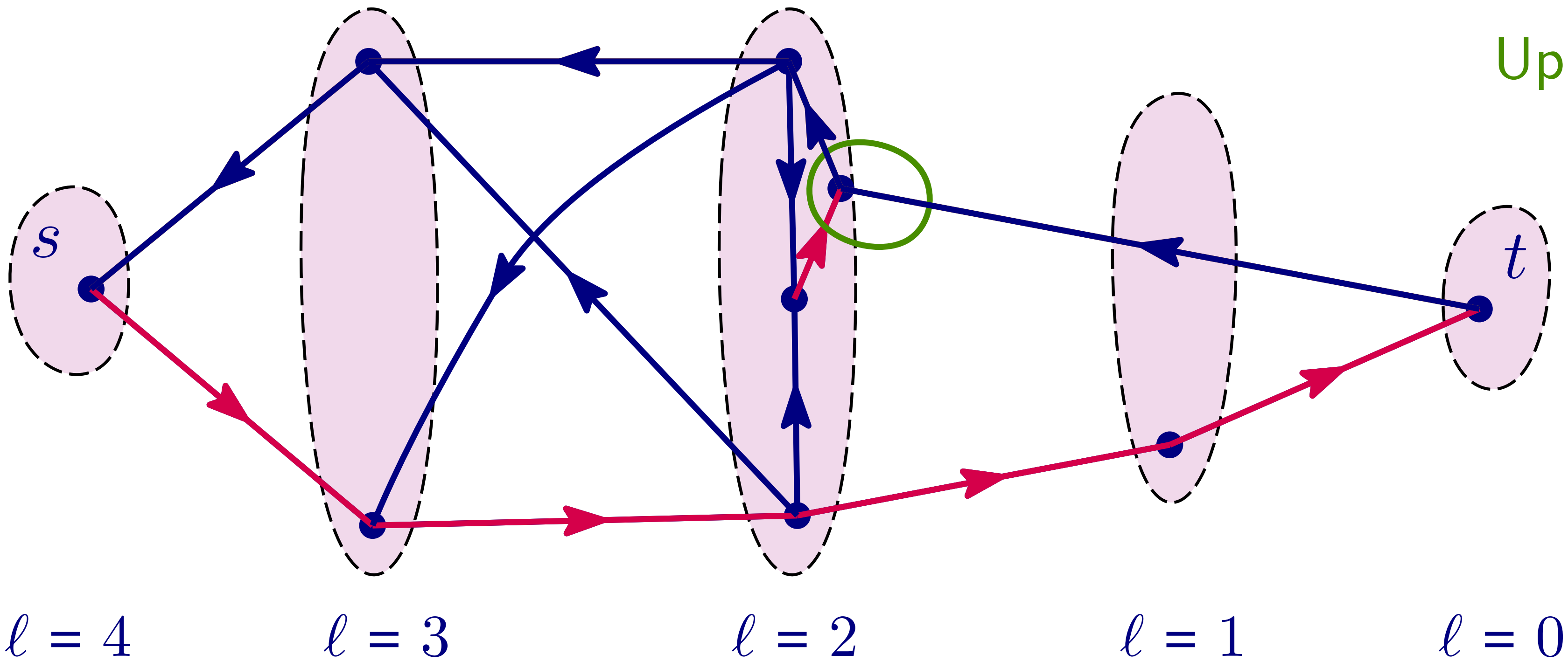
$\ell(v) = \mathrm{dist}(v, t)$

edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + 1$

Reverse it

Update ℓ?

Only increases!



RELABEL(v)
If no admissible out-edge:
$\ell(v) \leftarrow \ell(v) + 1$

Shortest Augmenting Path: follow admissible edges from s

Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88]

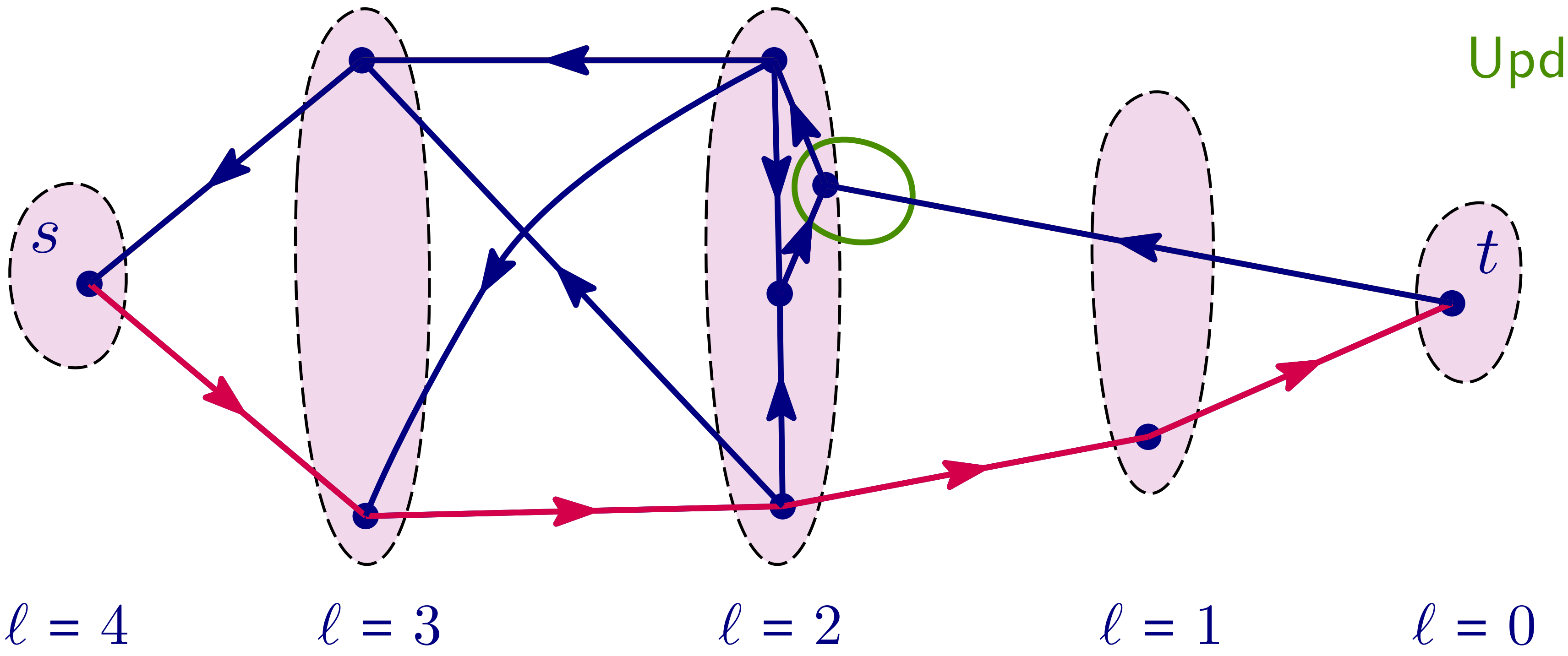
$\ell(v) = \mathrm{dist}(v, t)$

edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + 1$

Reverse it

Update ℓ?

Only increases!



RELABEL(v)
If no admissible out-edge:
$\ell(v) \leftarrow \ell(v) + 1$

Shortest Augmenting Path: follow admissible edges from s

Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88]

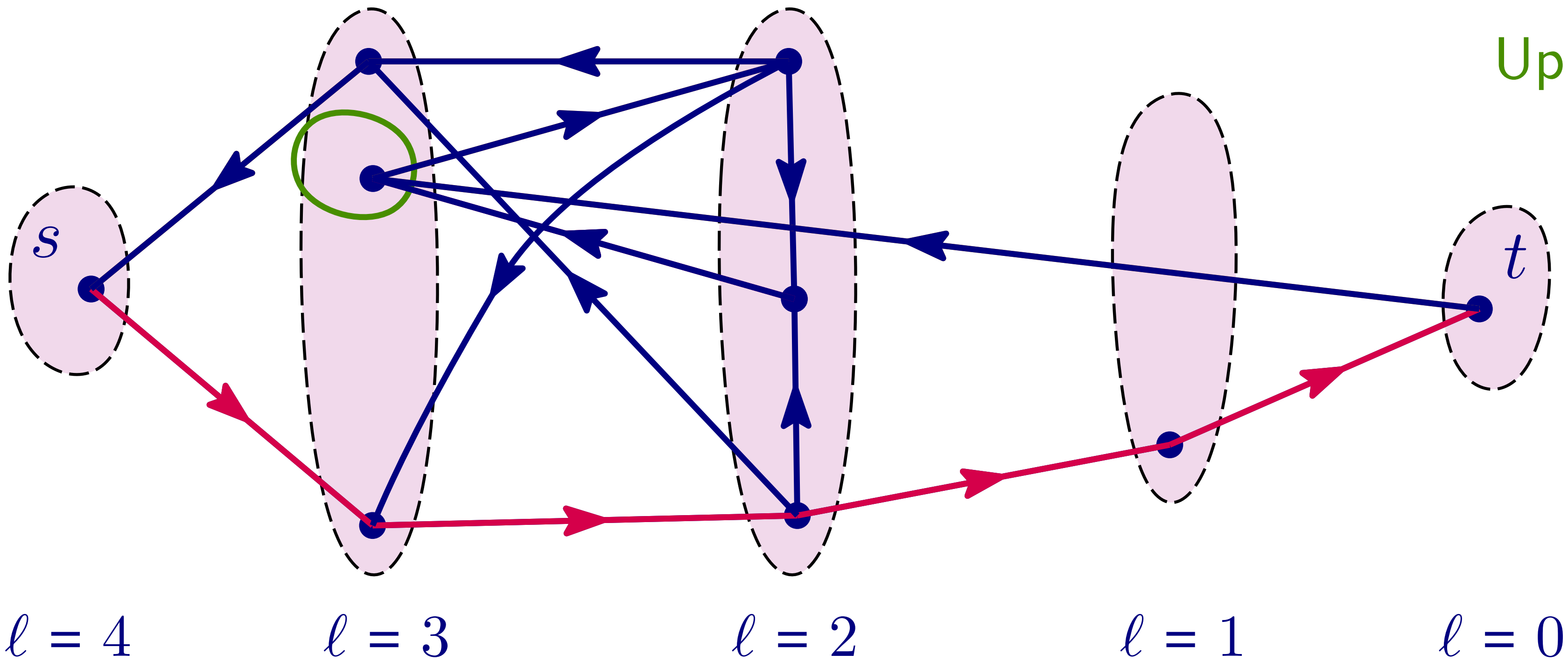
$\ell(v) = \text{dist}(v, t)$

edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + 1$

Reverse it

Update ℓ?

Only increases!



Shortest Augmenting Path: follow admissible edges from s

RELABEL(v)
If no admissible out-edge:
$\ell(v) \leftarrow \ell(v) + 1$

Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88]

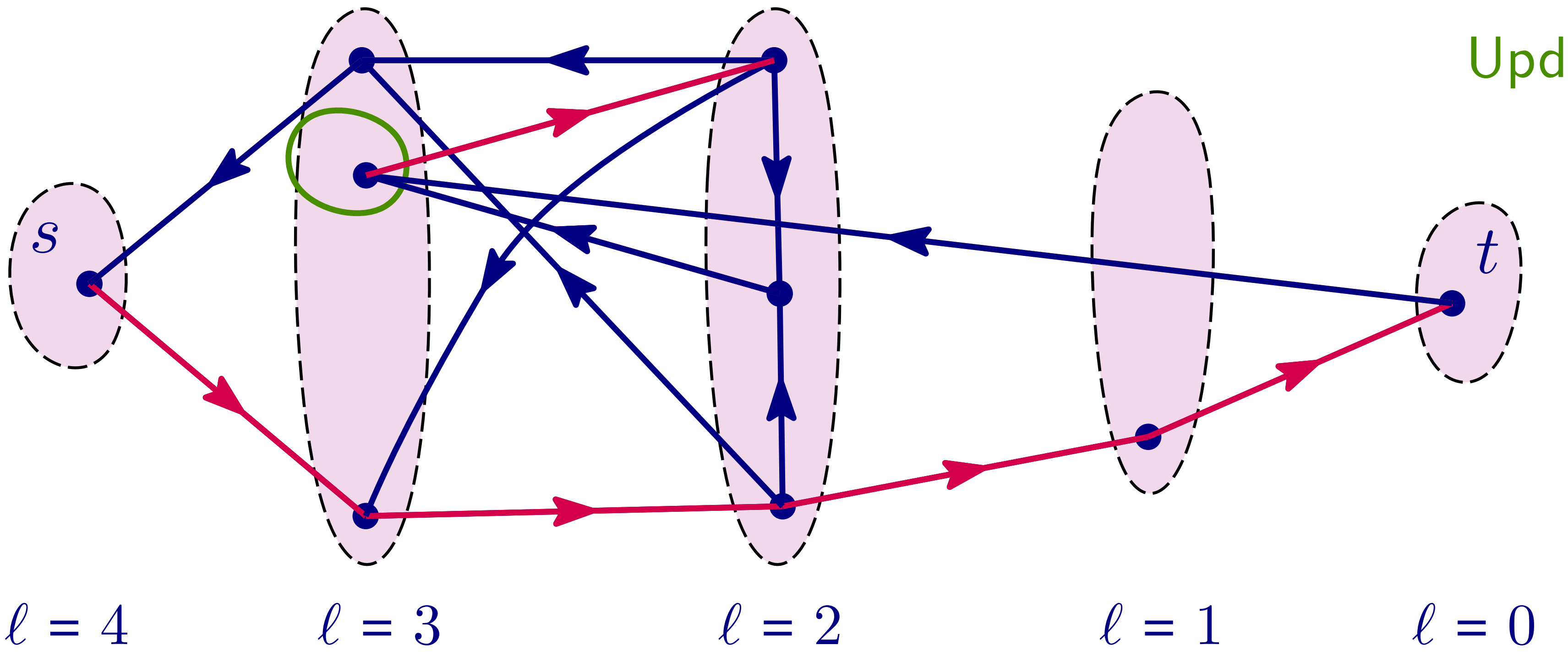
$\ell(v) = \mathrm{dist}(v, t)$

edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + 1$

Reverse it

Update ℓ?

Only increases!



Shortest Augmenting Path: follow admissible edges from s

RELABEL(v)
If no admissible out-edge:
$\ell(v) \leftarrow \ell(v) + 1$

Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88]

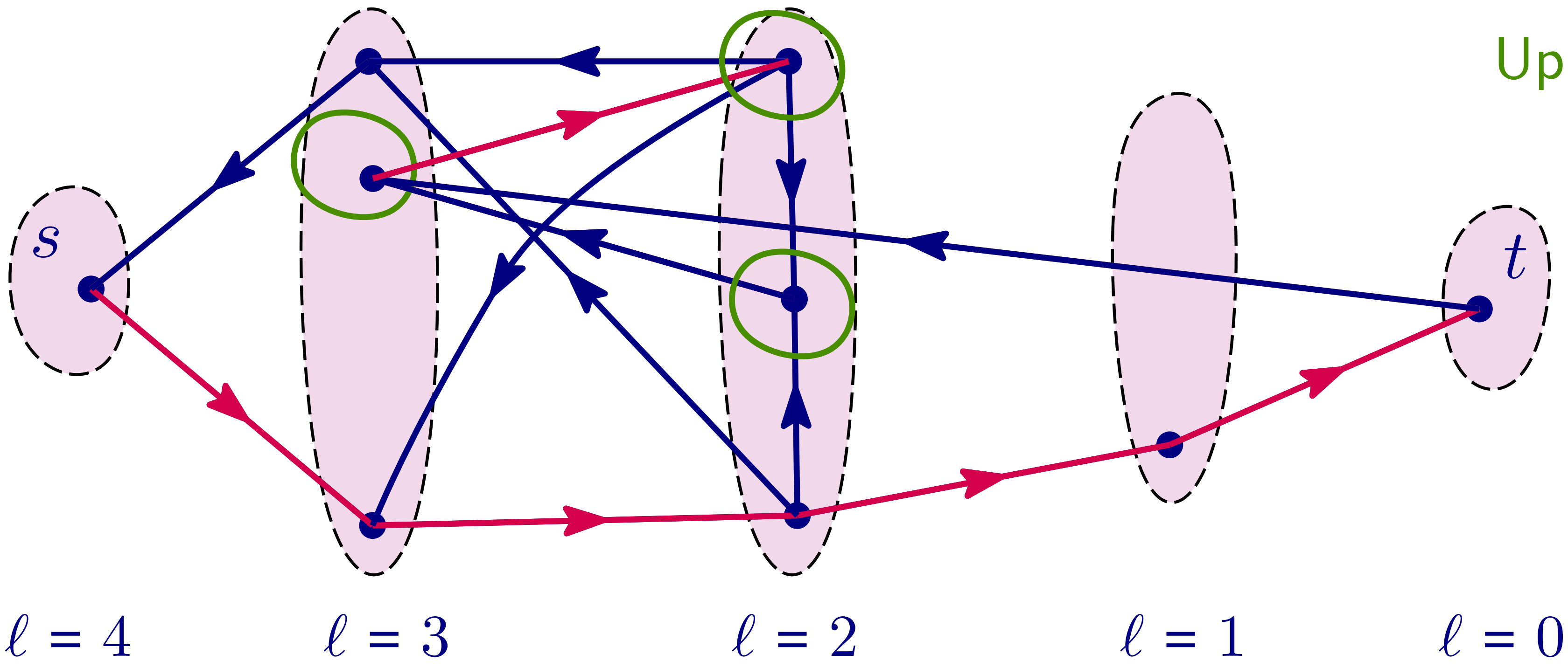
$\ell(v) = \mathrm{dist}(v, t)$

edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + 1$

Reverse it

Update ℓ?

Only increases!



Shortest Augmenting Path: follow admissible edges from s

RELABEL(v)
If no admissible out-edge:
$\ell(v) \leftarrow \ell(v) + 1$

Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88]

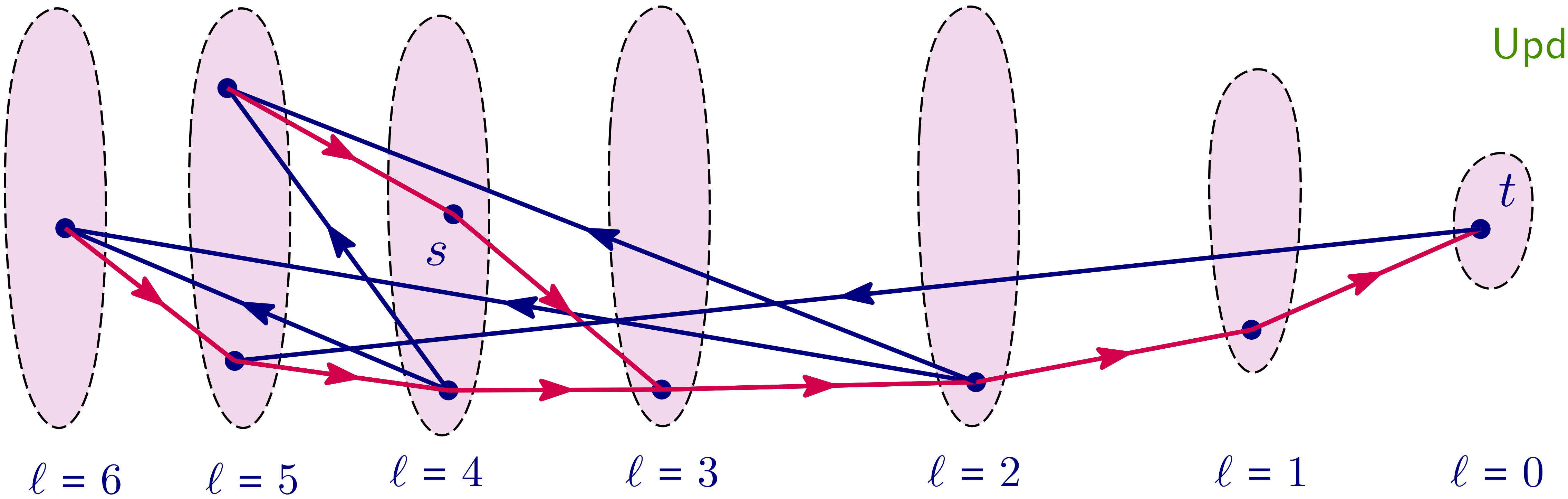
edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + 1$

$\ell(v) = \mathrm{dist}(v, t)$

Reverse it

Update ℓ?

Only increases!



Shortest Augmenting Path: follow admissible edges from s

RELABEL(v)
If no admissible out-edge:
$\ell(v) \leftarrow \ell(v) + 1$

Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88]

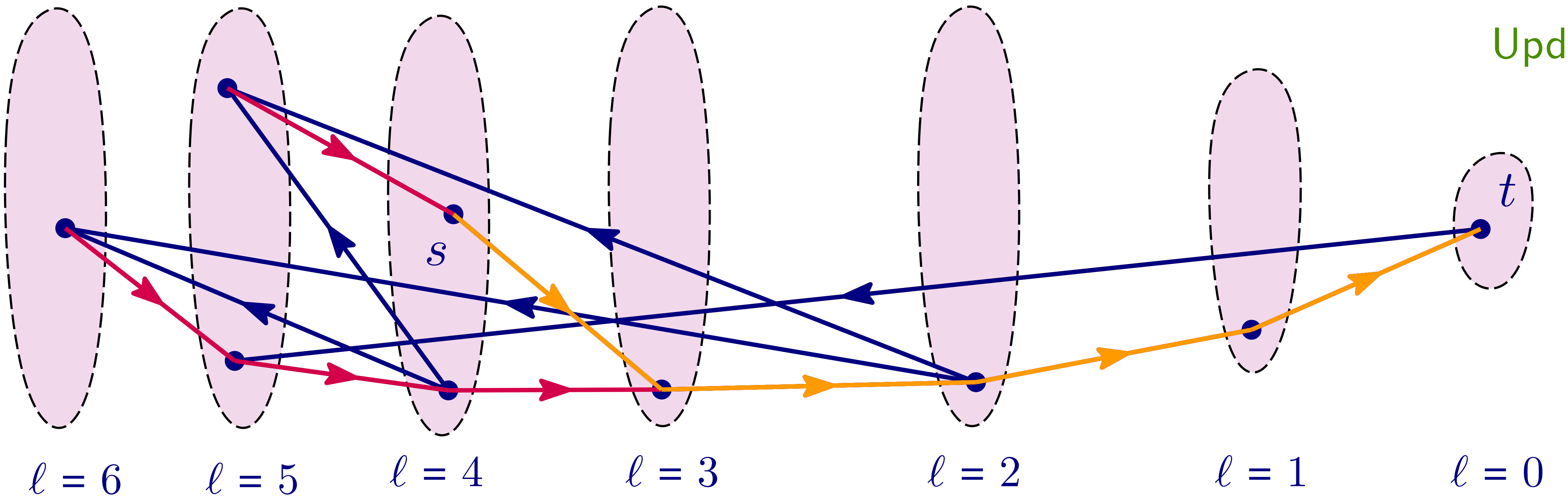
edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + 1$

$\ell(v) = \text{dist}(v, t)$

Reverse it

Update ℓ?

Only increases!



Shortest Augmenting Path: follow admissible edges from s

RELABEL(v)
If no admissible out-edge:
$\ell(v) \leftarrow \ell(v) + 1$

Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88] —Analysis

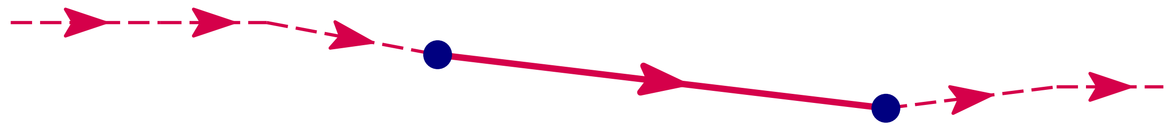
RELABEL $\quad O(n^2) \quad$ (n vertices, n layers)

Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88] —Analysis

RELABEL	$O(n^2)$	(n vertices, n layers)
Keeping track of admissible edges:	$O(nm)$	(after relabel: recheck incident edges)

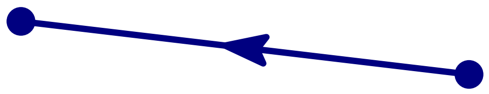
Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88] —Analysis

Relabel	$O(n^2)$	(n vertices, n layers)
Keeping track of admissible edges:	$O(nm)$	(after relabel: recheck incident edges)
Augmentations	$O(nm)$	(n per edge)



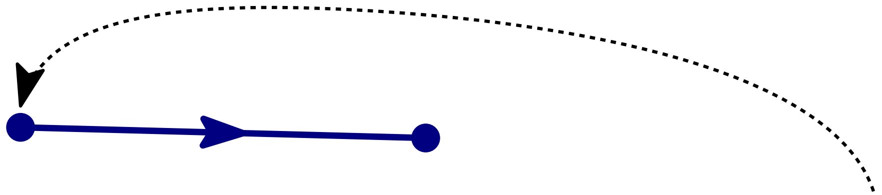
Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88] —Analysis

RELABEL	$O(n^2)$	(n vertices, n layers)
Keeping track of admissible edges:	$O(nm)$	(after relabel: recheck incident edges)
Augmentations	$O(nm)$	(n per edge)



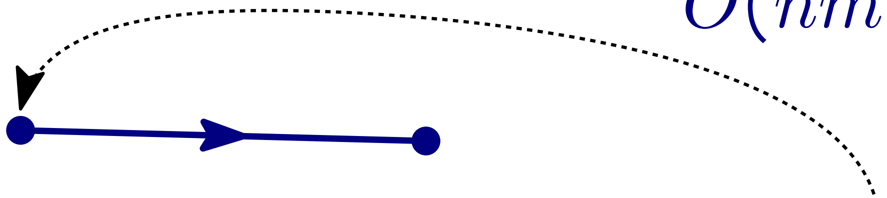
Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88] —Analysis

RELABEL	$O(n^2)$	(n vertices, n layers)
Keeping track of admissible edges:	$O(nm)$	(after relabel: recheck incident edges)
Augmentations	$O(nm)$	(n per edge)



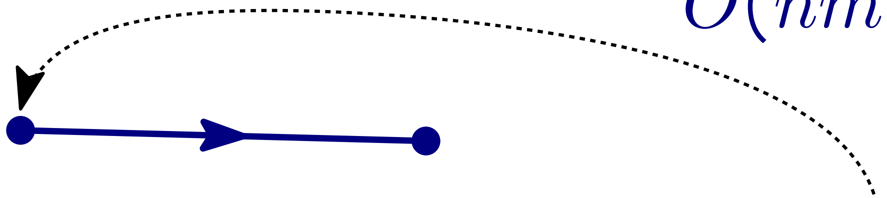
Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88] —Analysis

RELABEL	$O(n^2)$	(n vertices, n layers)
Keeping track of admissible edges:	$O(nm)$	(after relabel: recheck incident edges)
Augmentations	$O(nm)$	(n per edge)
	$O(nm \log n)$	(capacitated graphs: Link-Cut trees of admissible edges)



Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88] —Analysis

RELABEL	$O(n^2)$	(n vertices, n layers)
Keeping track of admissible edges:	$O(nm)$	(after relabel: recheck incident edges)
Augmentations	$O(nm)$	(n per edge)
	$O(nm \log n)$	(capacitated graphs: Link-Cut trees of admissible edges)



Total: $\tilde{O}(nm)$

How to speed it up?

New Idea: Edge Lengths

$\boldsymbol{w}(e) = 2$

$\boldsymbol{w}(e) = 10$

New Idea: Edge Lengths

"short" = "frequent"

$\boldsymbol{w}(e) = 2$

"long" = "infrequent"

$\boldsymbol{w}(e) = 10$

New Idea: Edge Lengths

"short" = "frequent" $\boldsymbol{w}(e) = 2$

"long" = "infrequent" $\boldsymbol{w}(e) = 10$

Guarantee: Path P in maxflow $\boldsymbol{f}^\star$

few long edges, potentially many short

New Idea: Edge Lengths

"short" = "frequent"

$\boldsymbol{w}(e) = 2$

"long" = "infrequent"

$\boldsymbol{w}(e) = 10$

Guarantee: Path P in maxflow $\boldsymbol{f}^{\star}$

$$\boldsymbol{w}(P) \leq n^{1+o(1)}$$

few long edges, potentially many short

New Idea: Edge Lengths

"short" = "frequent"

$\boldsymbol{w}(e) = 2$

"long" = "infrequent"

$\boldsymbol{w}(e) = 10$

Potential Faster Algo:
look for short paths

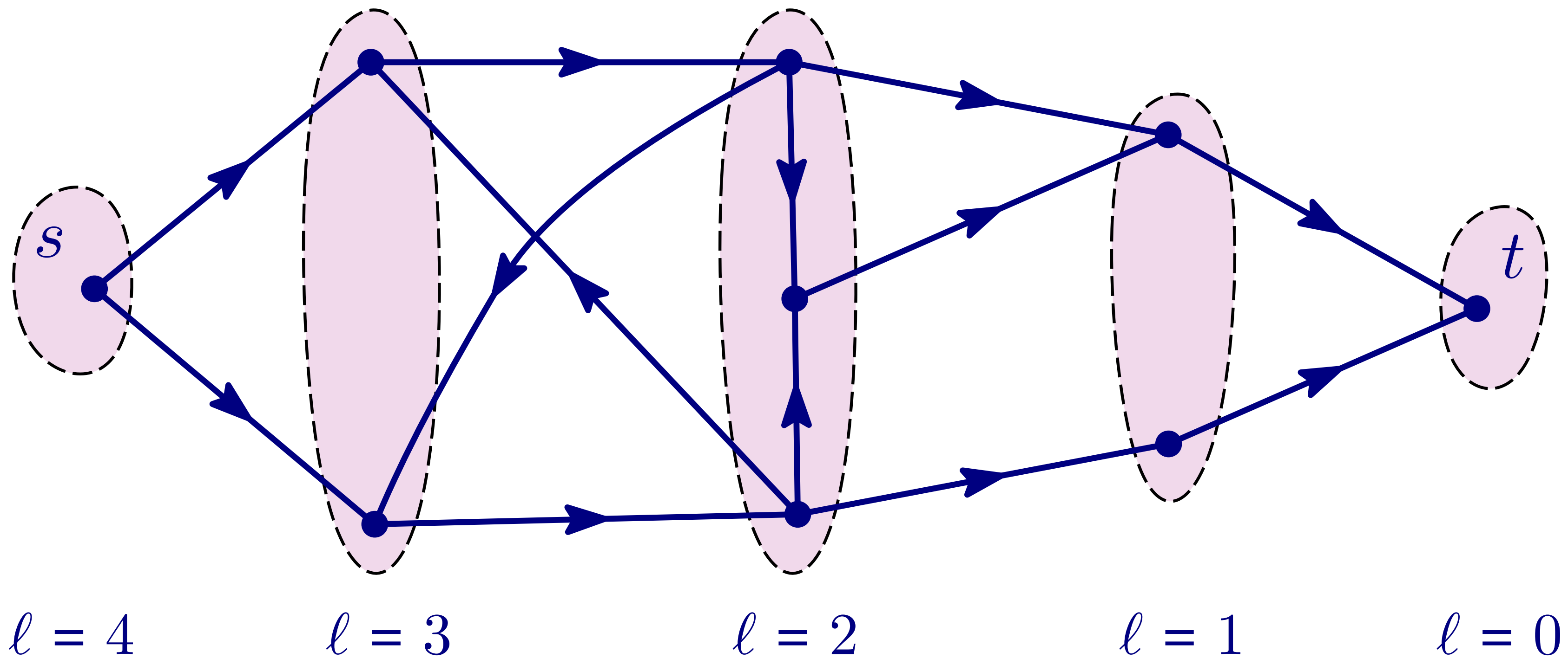
Guarantee: Path P in maxflow $\boldsymbol{f}^{\star}$

$\boldsymbol{w}(P) \leq n^{1+o(1)}$

few long edges, potentially many short

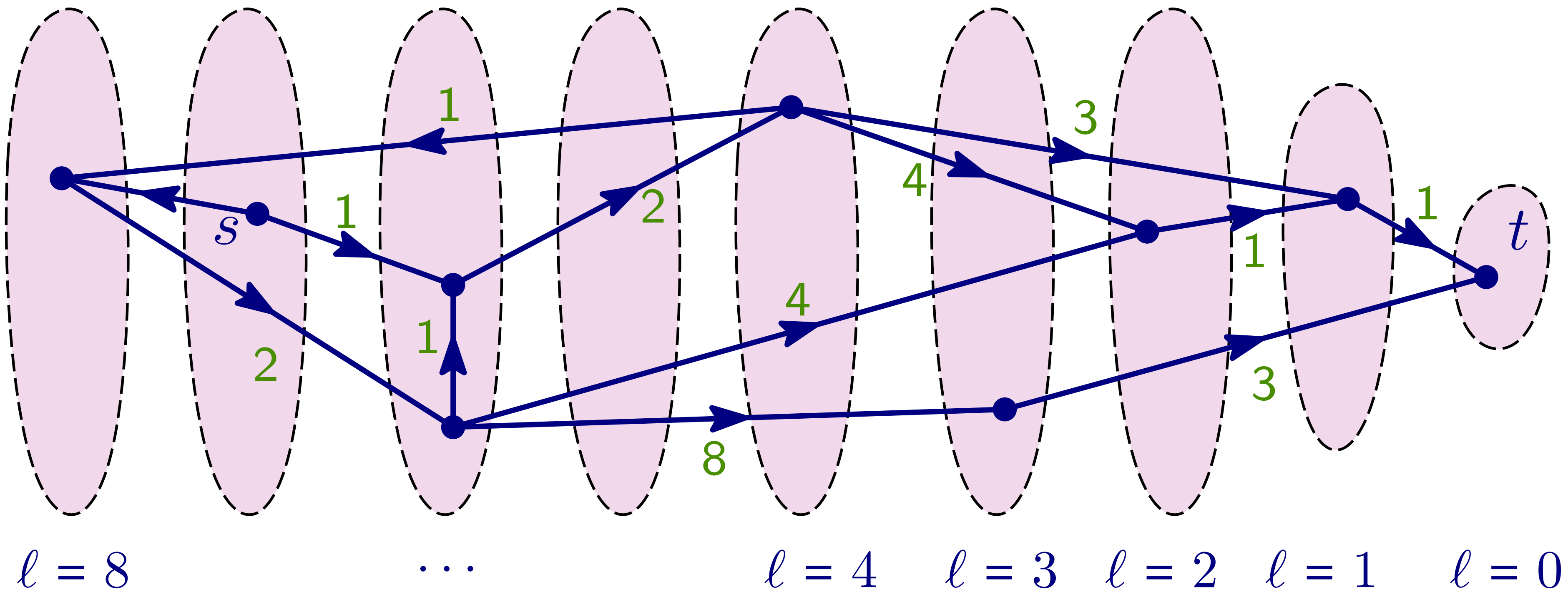
Weighted Push-Relabel

$\ell(v) = \text{dist}(v, t)$



Weighted Push-Relabel

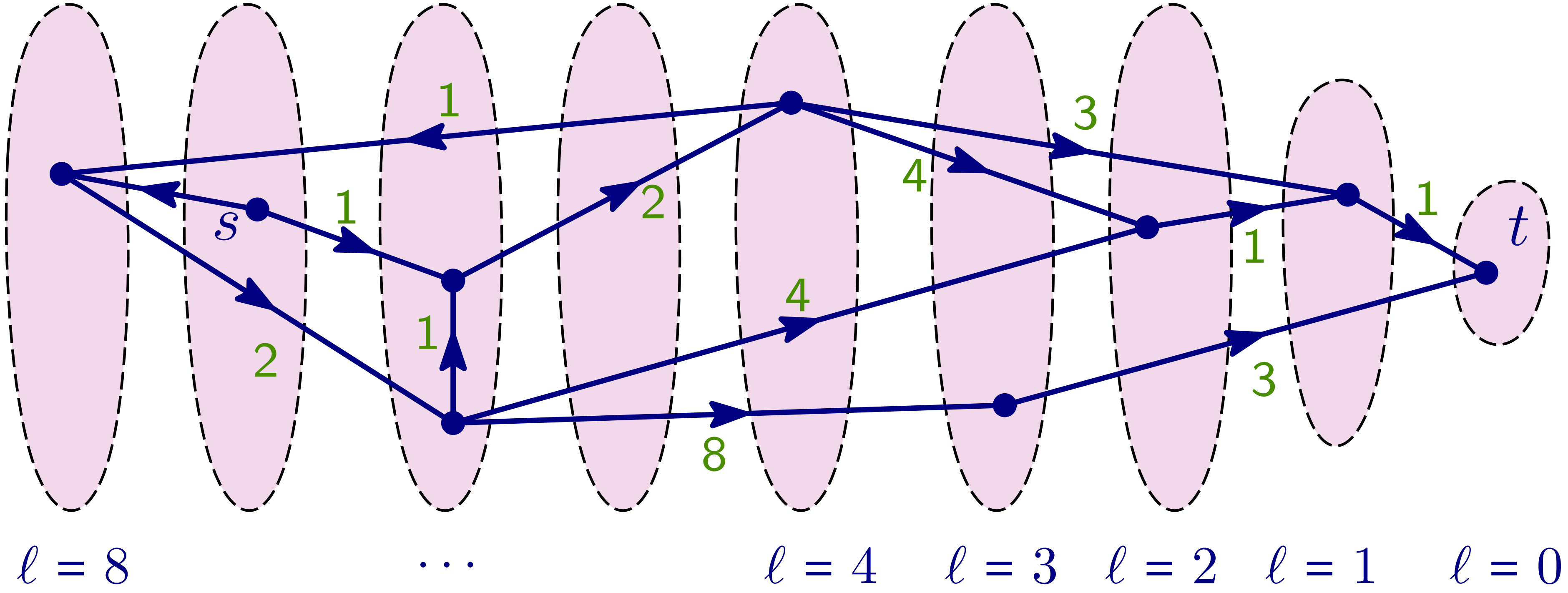
$\ell(v) = \mathrm{dist}_{\boldsymbol{w}}(v, t)$



Weighted Push-Relabel

$\ell(v) = \text{dist}_{\boldsymbol{w}}(v, t)$

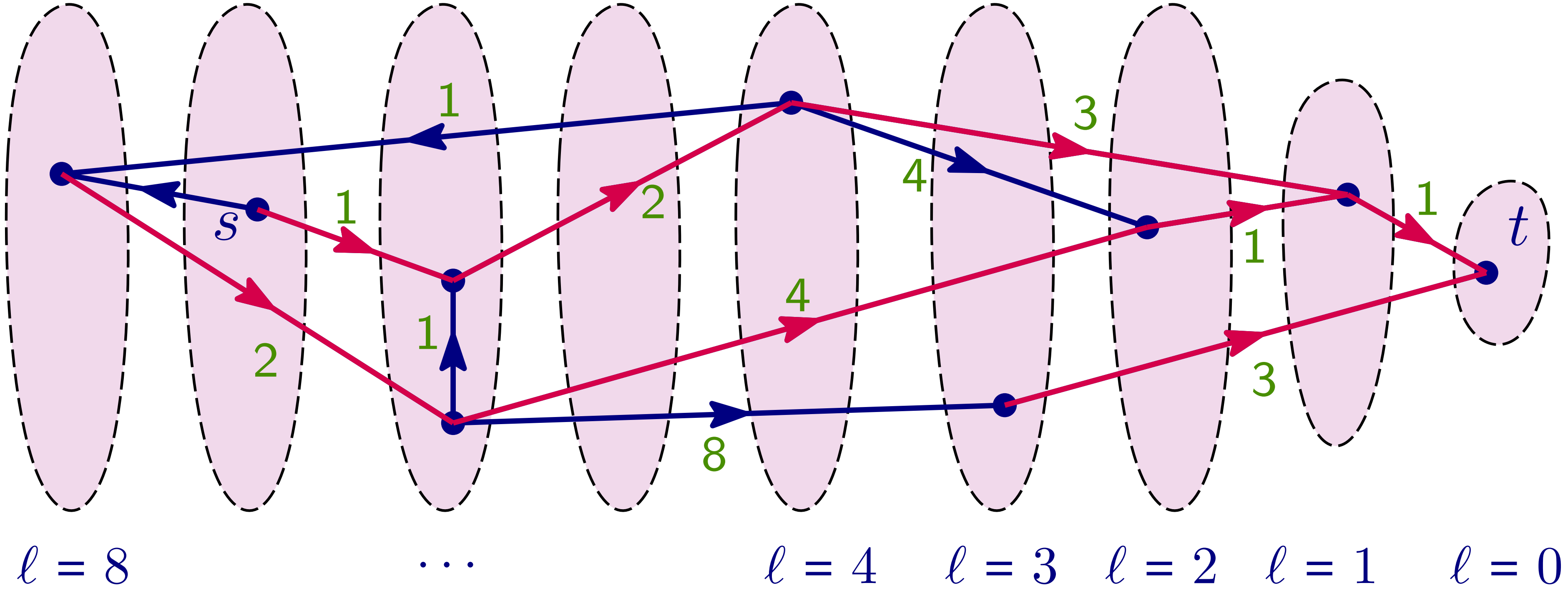
edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + 1$



Weighted Push-Relabel

$\ell(v) = \text{dist}_{\boldsymbol{w}}(v, t)$

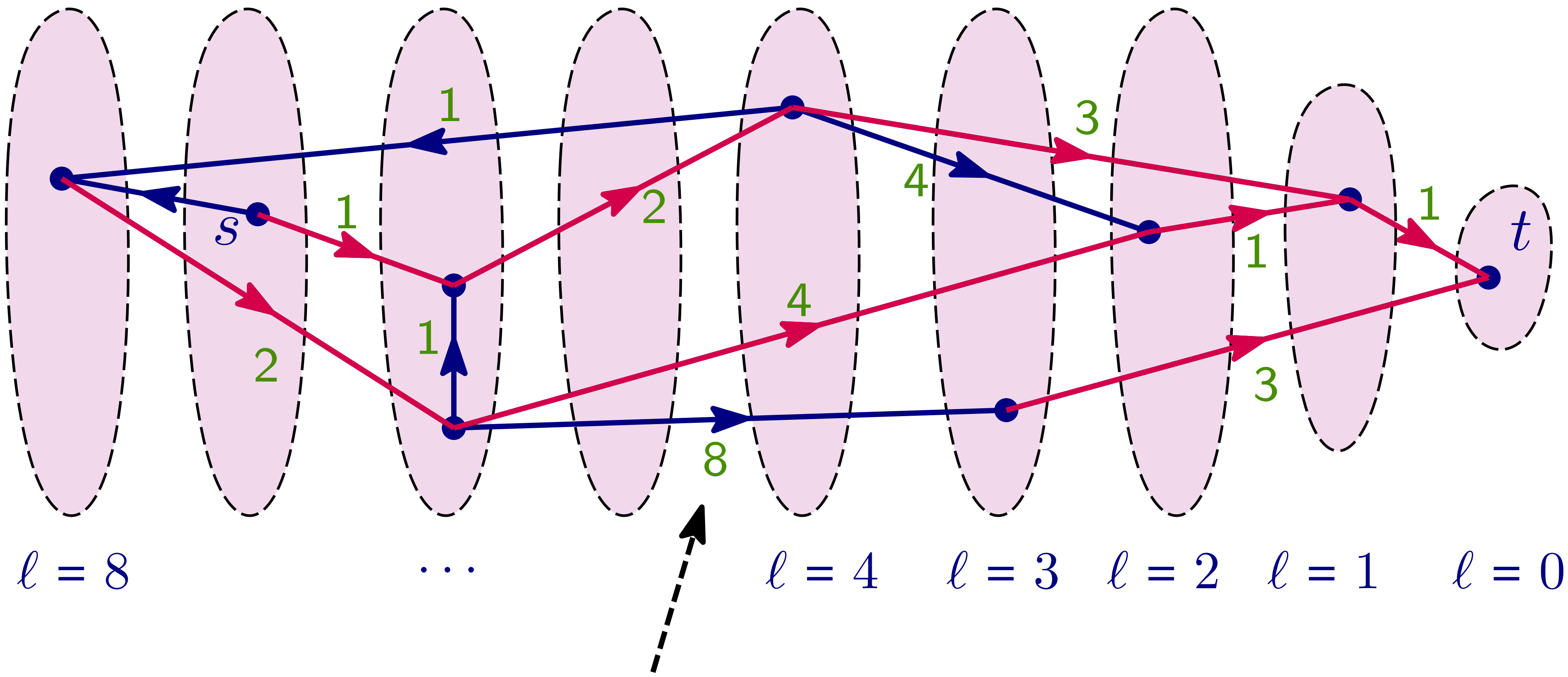
edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + \boldsymbol{w}(e)$



Weighted Push-Relabel

$\ell(v) = \mathrm{dist}_{\boldsymbol{w}}(v, t)$

edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + \boldsymbol{w}(e)$

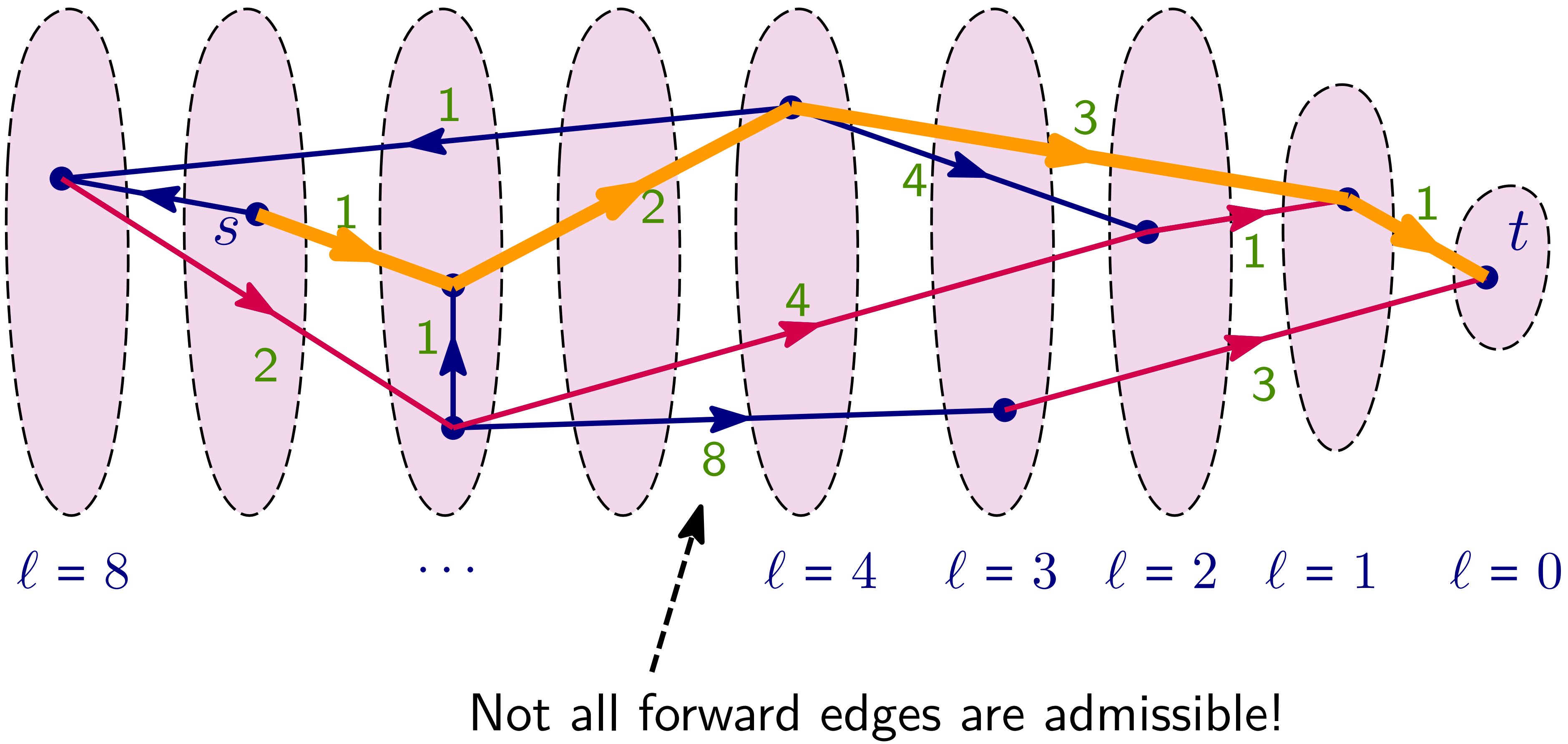


Not all forward edges are admissible!

Weighted Push-Relabel

$\ell(v) = \text{dist}_{\boldsymbol{w}}(v, t)$

edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + \boldsymbol{w}(e)$

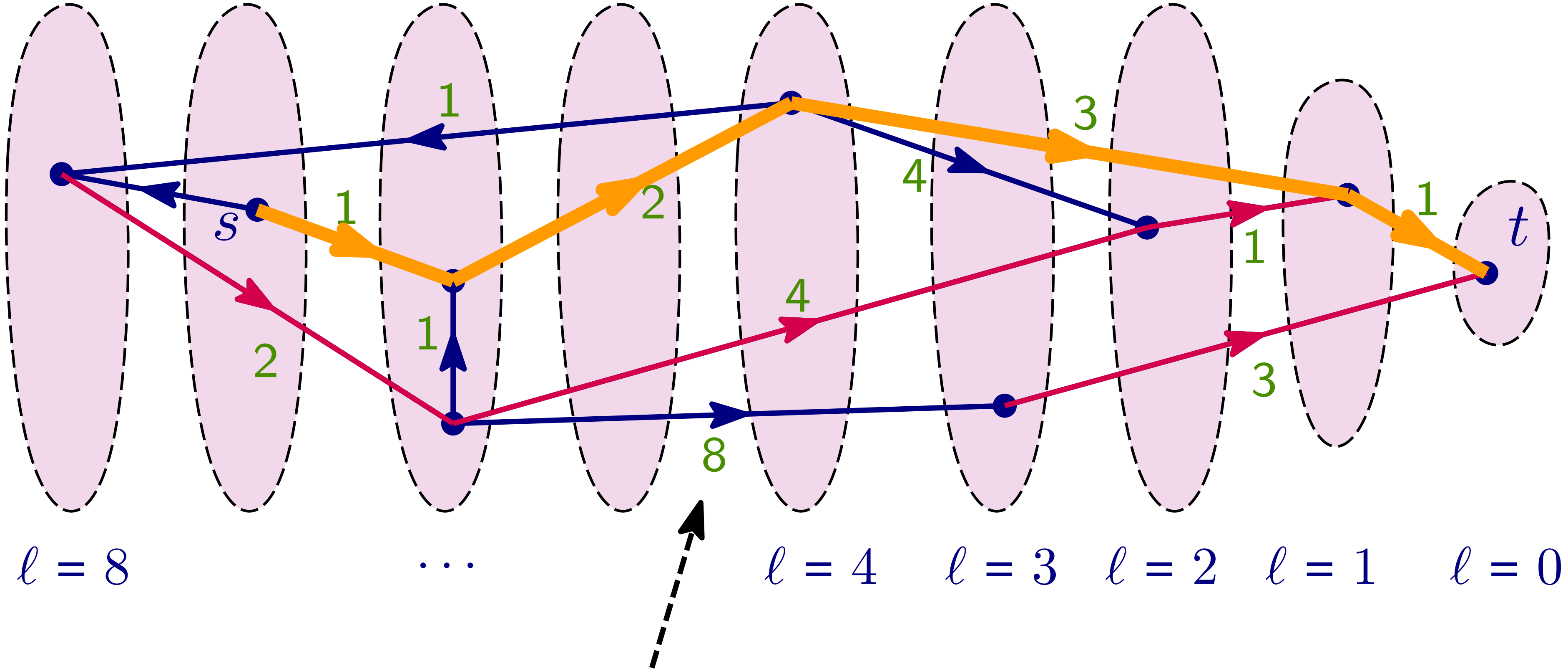


$\boldsymbol{w}$-Shortest Augmenting Path: follow admissible edges from s

Weighted Push-Relabel

$\ell(v) \approx \mathrm{dist}_{\boldsymbol{w}}(v,t)$

edge $e = (u,v)$ *admissible* iff $\ell(u) \approx \ell(v) + \boldsymbol{w}(e)$



Not all forward edges are admissible!

$\approx \boldsymbol{w}$-Shortest Augmenting Path: follow admissible edges from s

Running Time Analysis

RELABEL	$O(n^2)$	(n vertices, n layers)
Keeping track of admissible edges:	$O(nm)$	(after relabel: recheck incident edges)
Augmentations	$O(nm)$	(n per edge)
	$O(nm \log n)$	(capacitated graphs: Link-Cut trees of admissible edges)

Running Time Analysis

RELABEL	$O(n^2)$	(n vertices, n layers)
Keeping track of admissible edges:	$O(nm)$	(after relabel: recheck incident edges)
Augmentations	$O(nm)$	(n per edge)
	$O(nm \log n)$	(capacitated graphs: Link-Cut trees of admissible edges)

Running Time Analysis

RELABEL	$O(n^2)$	(n vertices, n layers)
Keeping track of admissible edges:	~~$O(nm)$~~	(after relabel: recheck ~~incident~~ edges)
Augmentations	~~$O(nm)$~~	(~~n~~ per edge)
	~~$O(nm \log n)$~~	(capacitated graphs: Link-Cut trees of admissible edges)

Goal: $\tilde{O}\left(\sum_{e \in E} \frac{\#\text{layers}}{\boldsymbol{w}(e)}\right)$

(#layers = $100n$, or $n^{1+o(1)}$)

Running Time Analysis

RELABEL	$O(n^2)$	(n vertices, n layers)
Keeping track of admissible edges:	~~$O(nm)$~~	(after relabel: recheck ~~incident~~ edges)
Augmentations	~~$O(nm)$~~	(~~n~~ per edge)
	~~$O(nm \log n)$~~	(capacitated graphs: Link-Cut trees of admissible edges)

Goal: $\tilde{O}\left(\sum_{e \in E} \frac{\#\text{layers}}{\boldsymbol{w}(e)}\right)$

(#layers = $100n$, or $n^{1+o(1)}$)

After relabel v: recheck only incident edges e where $\boldsymbol{w}(e)$ divides $\ell(v)$

Running Time Analysis

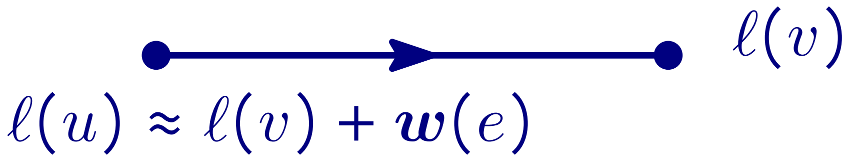
RELABEL	$O(n^2)$	(n vertices, n layers)
Keeping track of admissible edges:	$O(nm)$	(after relabel: recheck incident edges)
Augmentations	$O(nm)$	(n per edge)
	$O(nm \log n)$	(capacitated graphs: Link-Cut trees of admissible edges)

Goal: $\tilde{O}\left(\sum_{e \in E} \frac{\#\text{layers}}{\boldsymbol{w}(e)}\right)$

(#layers = $100n$, or $n^{1+o(1)}$)

After relabel v: recheck only incident edges e where $\boldsymbol{w}(e)$ divides $\ell(v)$

Augmentations



Running Time Analysis

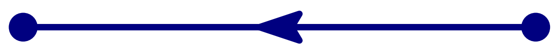
Relabel	$O(n^2)$	(n vertices, n layers)
Keeping track of admissible edges:	$O(nm)$	(after relabel: recheck ~~incident~~ edges)
Augmentations	$O(nm)$	(~~n~~ per edge)
	$O(nm\log n)$	(capacitated graphs: Link-Cut trees of admissible edges)

Goal: $\tilde{O}\left(\sum_{e\in E}\frac{\#\text{layers}}{\boldsymbol{w}(e)}\right)$

(#layers = $100n$, or $n^{1+o(1)}$)

After relabel v: recheck only incident edges e where $\boldsymbol{w}(e)$ divides $\ell(v)$

Augmentations



Running Time Analysis

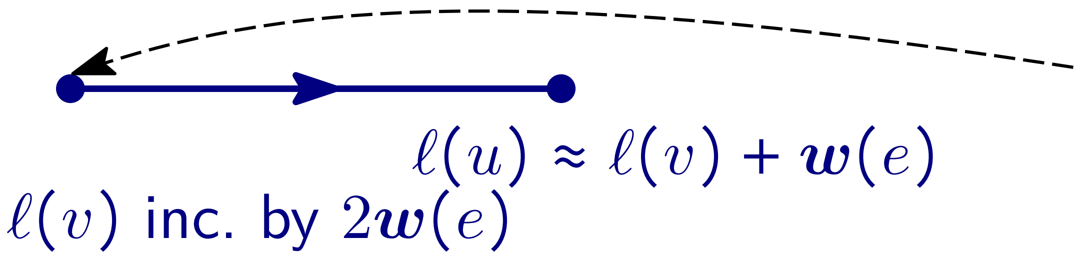
RELABEL	$O(n^2)$	(n vertices, n layers)
Keeping track of admissible edges:	~~$O(nm)$~~	(after relabel: recheck ~~incident~~ edges)
Augmentations	~~$O(nm)$~~	(~~n~~ per edge)
	~~$O(nm \log n)$~~	(capacitated graphs: Link-Cut trees of admissible edges)

Goal: $\tilde{O}\left(\sum_{e \in E} \frac{\#\text{layers}}{\boldsymbol{w}(e)}\right)$

(#layers = $100n$, or $n^{1+o(1)}$)

After relabel v: recheck only incident edges e where $\boldsymbol{w}(e)$ divides $\ell(v)$

Augmentations



Pseudo-Code

Algorithm 1: $\textsc{PushRelabel}(G, \boldsymbol{c}, \boldsymbol{\Delta}, \boldsymbol{\nabla}, \boldsymbol{w}, h)$

Initialize $\boldsymbol{f}$ as the empty flow.
Let $\boldsymbol{\ell}(v) = 0$ for all $v \in V$. `// levels`
Mark each edge $e \in \overrightarrow{E} \cup \overleftarrow{E}$ as *inadmissible* and all vertices as *alive*.

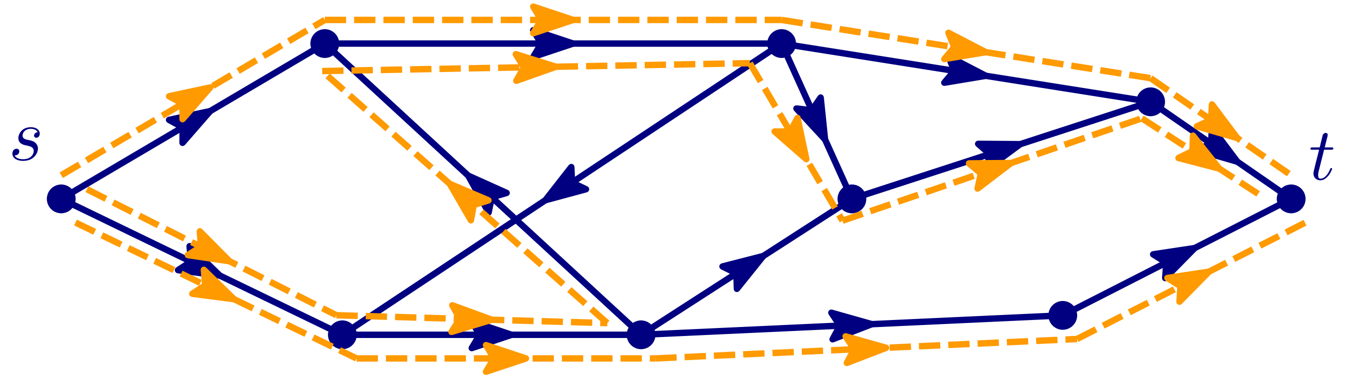
function $\textsc{Relabel}(v)$
  Set $\boldsymbol{\ell}(v) \leftarrow \boldsymbol{\ell}(v) + 1$.
  if $\boldsymbol{\ell}(v) > 9h$ **then**
    mark v as *dead* and **return**.
  for *each edge* $e \ni v$ *where* $\boldsymbol{w}(e)$ *divides* $\boldsymbol{\ell}(v)$ **do**
    Let $(x, y) = e$.
    if $\boldsymbol{\ell}(x) - \boldsymbol{\ell}(y) \geq 2\boldsymbol{w}(e)$ *and* $\boldsymbol{c_f}(e) > 0$ **then** mark e as *admissible*.
    else mark e as *inadmissible*.

main loop
  while *there is an alive vertex* v *with* $\boldsymbol{\nabla_f}(v) = 0$ *and without an admissible out-edge* **do**
    $\textsc{Relabel}(v)$
  if *there is some alive vertex* s *with* $\boldsymbol{\Delta_f}(s) > 0$ **then**
    `// P is an "augmenting path"`
    Trace a path P from s to some sink t, by arbitrarily following admissible out-edges.
    Let $c^{\text{augment}} \leftarrow \min\{\boldsymbol{\Delta_f}(s), \boldsymbol{\nabla_f}(t), \min_{e \in P} \boldsymbol{c_f}(e)\}$.
    for $e \in P$ **do** `// Augment f along P`
      if *e is a forward edge* **then** $\boldsymbol{f}(e) \leftarrow \boldsymbol{f}(e) + c^{\text{augment}}$.
      else $\boldsymbol{f}(e') \leftarrow \boldsymbol{f}(e') - c^{\text{augment}}$, where e' is the corresponding forward edge to e.
      Adjust residual capacities $\boldsymbol{c_f}$ of e and the corresponding reverse edge.
      if $\boldsymbol{c_f}(e) = 0$ **then** mark e as *inadmissible*.
    `//` $\boldsymbol{\Delta_f}(s)$ `and` $\boldsymbol{\nabla_f}(t)$ `goes down by` c^{augment}
  else return $\boldsymbol{f}$

Similar to normal Augment-Relabel / Push-Relabel

Good Edge Lengths

- *Good* $\boldsymbol{w}$:
  - $\sum_{e \in E} \frac{n}{\boldsymbol{w}(e)}$ is small $(\approx n^{2+o(1)}$, running-time)
  - "Optimal" flow $f^\star$ which is short w.r.t. $\boldsymbol{w}$ (flow paths of length $\approx n^{1+o(1)}$)

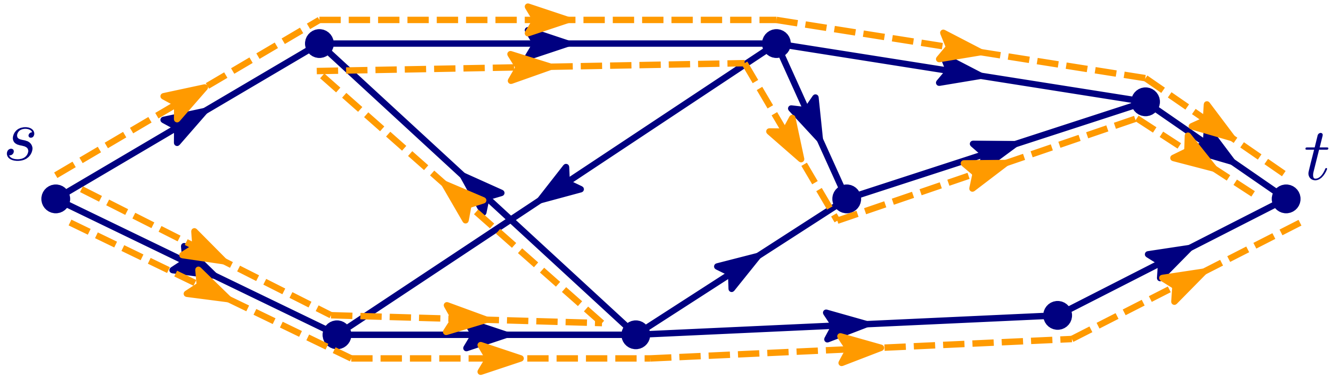


Good Edge Lengths

- *Good* $\boldsymbol{w}$:
  - $\sum_{e \in E} \frac{n}{\boldsymbol{w}(e)}$ is small (≈ $n^{2+o(1)}$, running-time)
  - "Optimal" flow $\boldsymbol{f}^\star$ which is short w.r.t. $\boldsymbol{w}$ (flow paths of length ≈ $n^{1+o(1)}$)

Lemma.

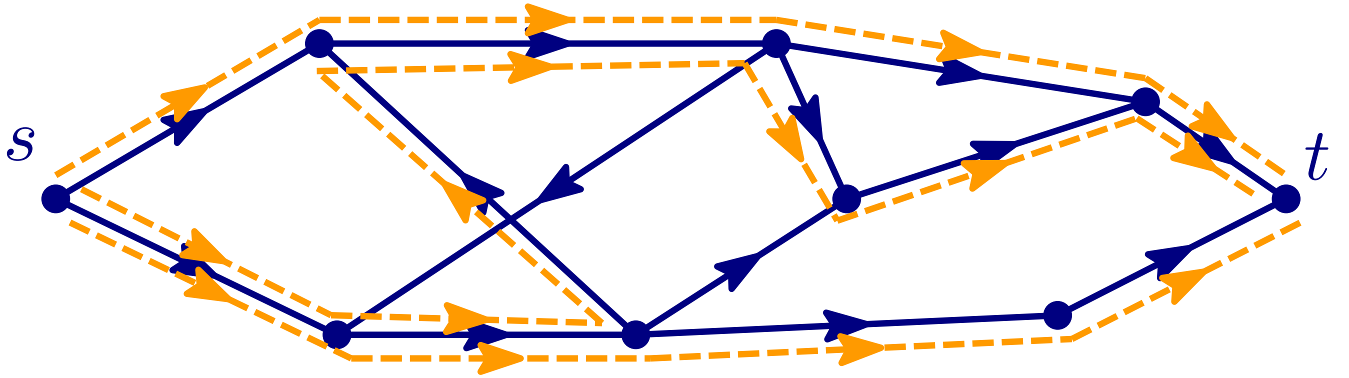
Weighted Push-Relabel finds $\boldsymbol{f}$ with $|\boldsymbol{f}| \geq \frac{1}{10}|\boldsymbol{f}^\star|$



Good Edge Lengths

- *Good* $\boldsymbol{w}$:
  - $\sum_{e \in E} \frac{n}{\boldsymbol{w}(e)}$ is small (≈ $n^{2+o(1)}$, running-time)
  - "Optimal" flow $\boldsymbol{f}^\star$ which is short w.r.t. $\boldsymbol{w}$ (flow paths of length ≈ $n^{1+o(1)}$)

Lemma.
Weighted Push-Relabel finds $\boldsymbol{f}$
with $|\boldsymbol{f}| \geq \frac{1}{10}|\boldsymbol{f}^\star|$

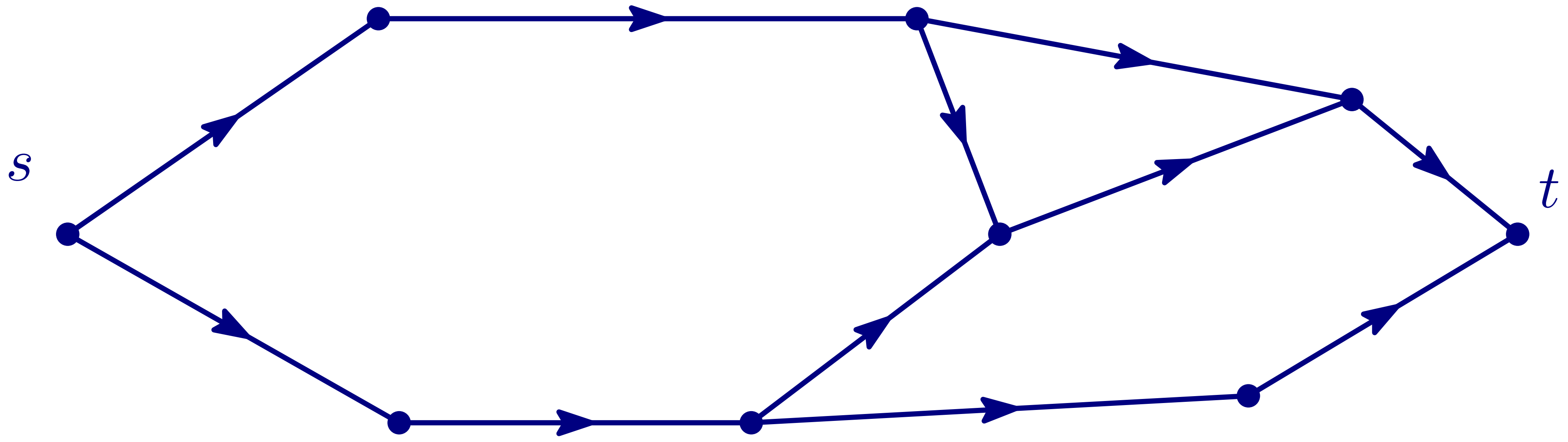


Proof Sketch.
If not: $|\boldsymbol{f}| < \frac{1}{10}|\boldsymbol{f}^\star|$
$\implies$ some flow path is still short in residual graph $G_{\boldsymbol{f}}$

How to find good edge lengths?

Directed Acyclic Graphs (DAG)

Def: no directed cycles

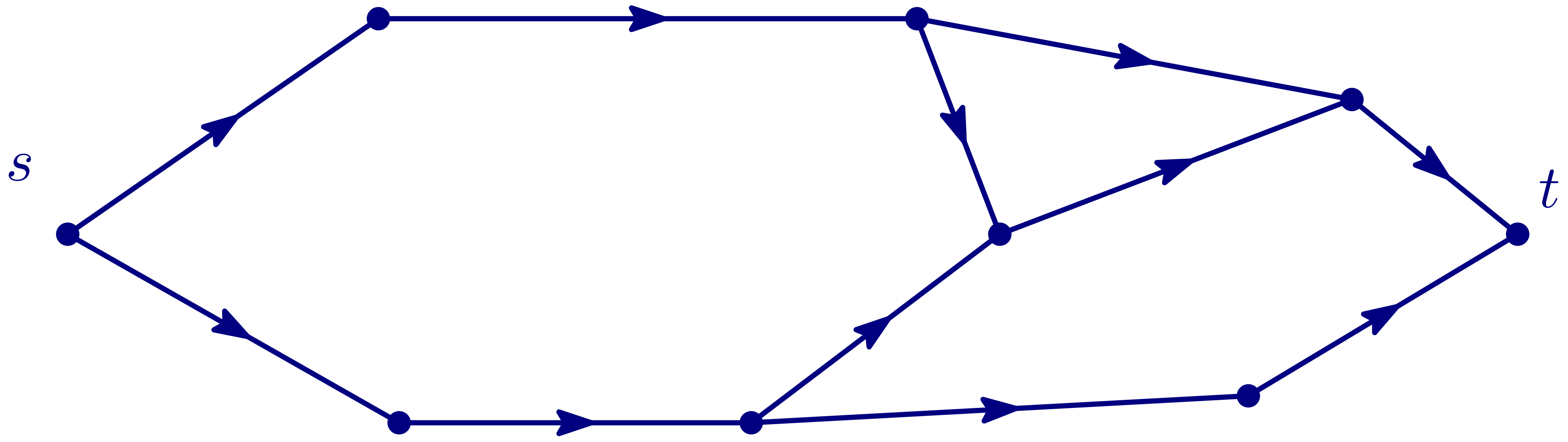


Directed Acyclic Graphs (DAG)

Def: no directed cycles

Good edge lenghts $\boldsymbol{w}$?

- $\sum_{e \in E} \frac{n}{\boldsymbol{w}(e)}$ is small
- "Optimal" flow $\boldsymbol{f}^{\star}$ which is short w.r.t. $\boldsymbol{w}$



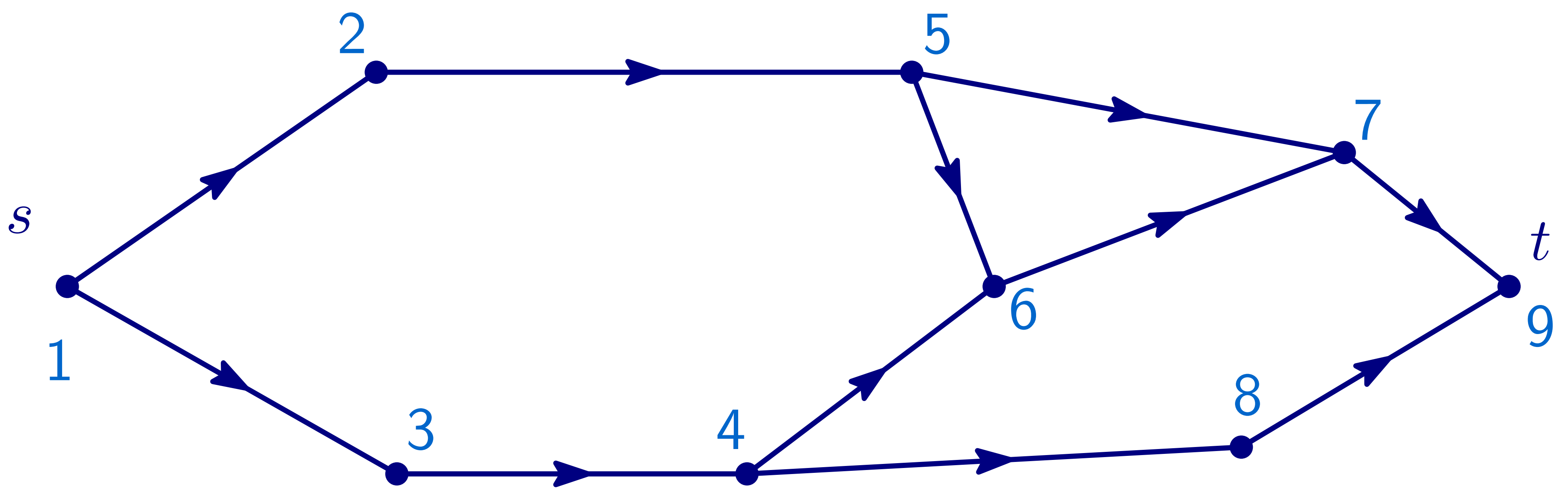
Directed Acyclic Graphs (DAG)

Def: no directed cycles

Topological order τ

Good edge lenghts w?

- $\sum_{e \in E} \frac{n}{w(e)}$ is small
- "Optimal" flow $f^\star$ which is short w.r.t. w



Directed Acyclic Graphs (DAG)

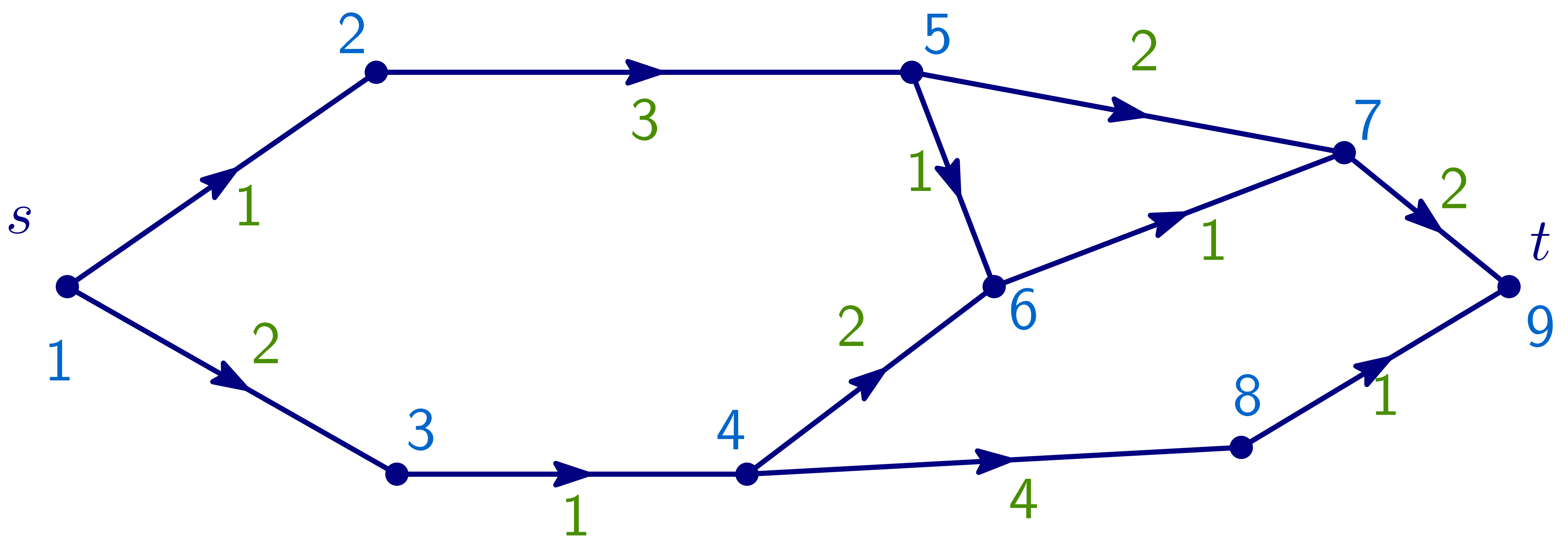
Def: no directed cycles

Topological order $\boldsymbol{\tau}$

$$\boldsymbol{w}(\boldsymbol{u},\boldsymbol{v}) = |\boldsymbol{\tau}(u) - \boldsymbol{\tau}(v)|$$

Good edge lenghts $\boldsymbol{w}$?

- $\sum_{e \in E} \frac{n}{\boldsymbol{w}(e)}$ is small
- "Optimal" flow $\boldsymbol{f}^\star$ which is short w.r.t. $\boldsymbol{w}$



Directed Acyclic Graphs (DAG)

Def: no directed cycles

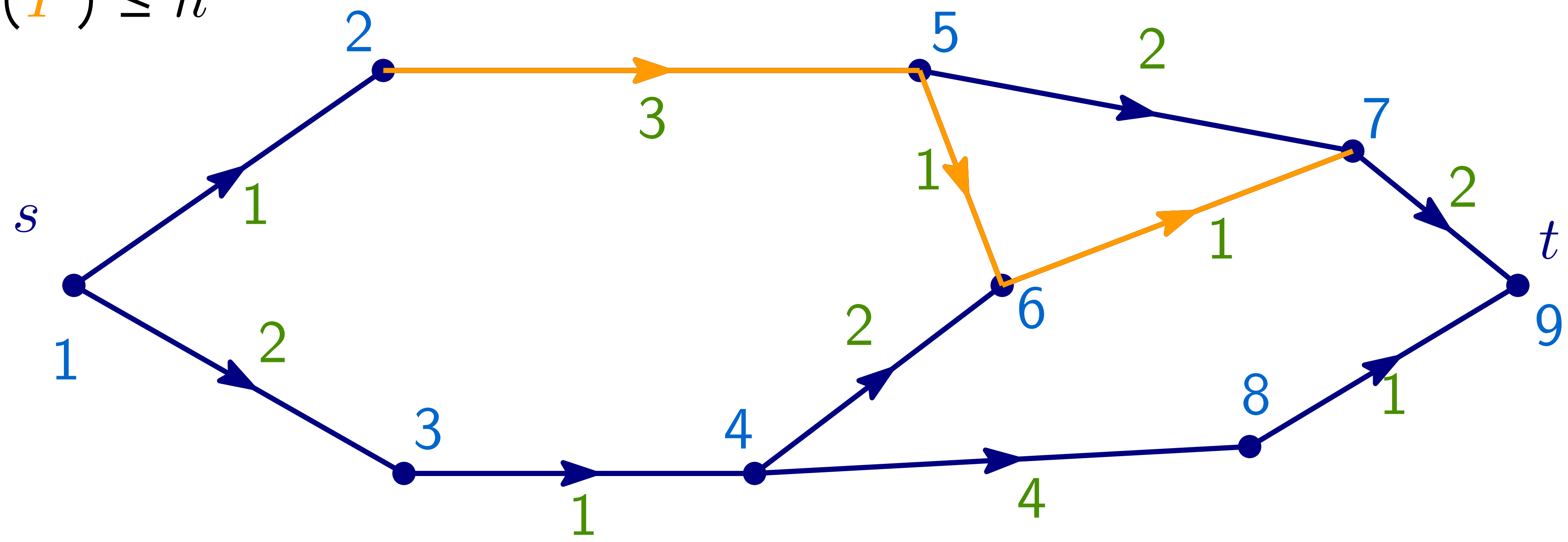
Topological order $\boldsymbol{\tau}$

$$\boldsymbol{w}(\boldsymbol{u},\boldsymbol{v}) = |\boldsymbol{\tau}(u) - \boldsymbol{\tau}(v)|$$

$$\boldsymbol{w}(P) \leq n$$

Good edge lenghts $\boldsymbol{w}$?

- $\sum_{e \in E} \frac{n}{\boldsymbol{w}(e)}$ is small
- "Optimal" flow $f^{\star}$ which is short w.r.t. $\boldsymbol{w}$



Directed Acyclic Graphs (DAG)

Def: no directed cycles

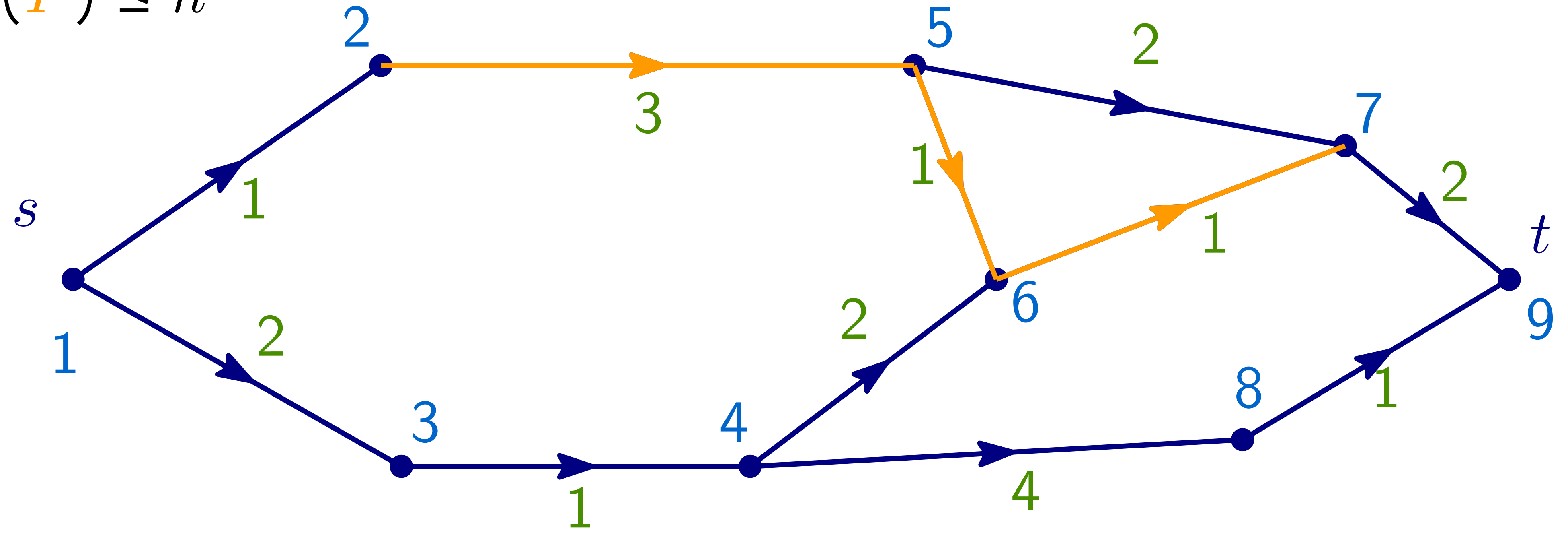
Topological order $\boldsymbol{\tau}$

$\boldsymbol{w}(\boldsymbol{u}, \boldsymbol{v}) = |\boldsymbol{\tau}(u) - \boldsymbol{\tau}(v)|$

$\boldsymbol{w}(P) \leq n$

Good edge lenghts $\boldsymbol{w}$?

- $\sum_{e \in E} \frac{n}{\boldsymbol{w}(e)}$ is small
- "Optimal" flow $f^\star$ which is short w.r.t. $\boldsymbol{w}$



$\sum_{(u,v)} \frac{n}{|\boldsymbol{\tau}(u) - \boldsymbol{\tau}(v)|} \leq n \sum_{v \in V} \sum_{k=1}^{n-1} \frac{1}{k} \leq n^2 \log n$

Directed Acyclic Graphs (DAG)

Def: no directed cycles

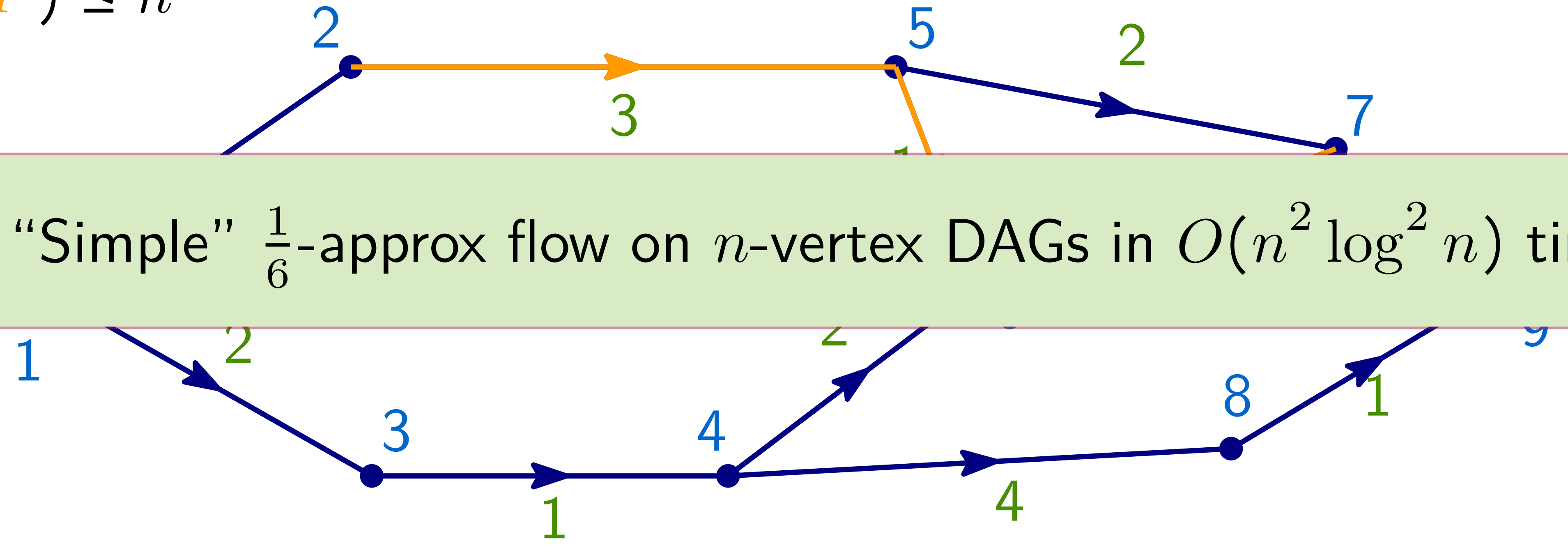
Topological order $\boldsymbol{\tau}$

$\boldsymbol{w(u,v)} = |\boldsymbol{\tau}(u) - \boldsymbol{\tau}(v)|$

$\boldsymbol{w}(P) \le n$

Good edge lenghts $\boldsymbol{w}$?

- $\sum_{e \in E} \frac{n}{\boldsymbol{w}(e)}$ is small
- "Optimal" flow $f^\star$ which is short w.r.t. $\boldsymbol{w}$



Theorem: "Simple" $\frac{1}{6}$-approx flow on n-vertex DAGs in $O(n^2 \log^2 n)$ time.

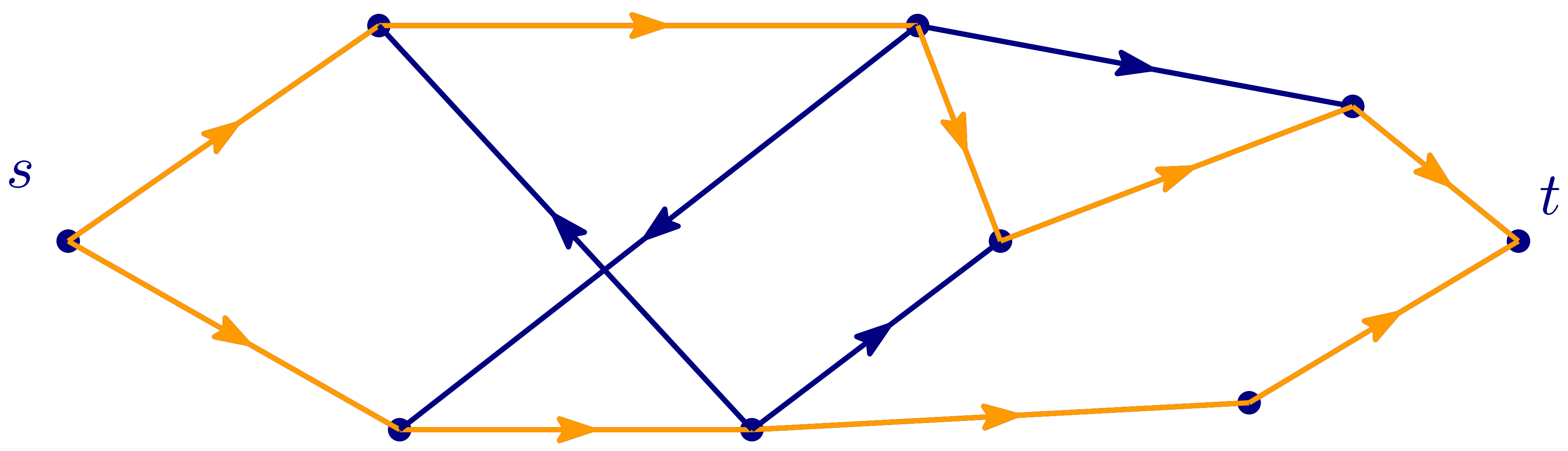
$$\sum_{(u,v)} \frac{n}{|\boldsymbol{\tau}(u) - \boldsymbol{\tau}(v)|} \le n \sum_{v \in V} \sum_{k=1}^{n-1} \frac{1}{k} \le n^2 \log n$$

General Graphs — Attempt One

1. Compute maxflow $f^\star$

Good edge lenghts $\boldsymbol{w}$?

- $\sum_{e \in E} \frac{n}{\boldsymbol{w}(e)}$ is small
- "Optimal" flow $f^\star$ which is short w.r.t. $\boldsymbol{w}$

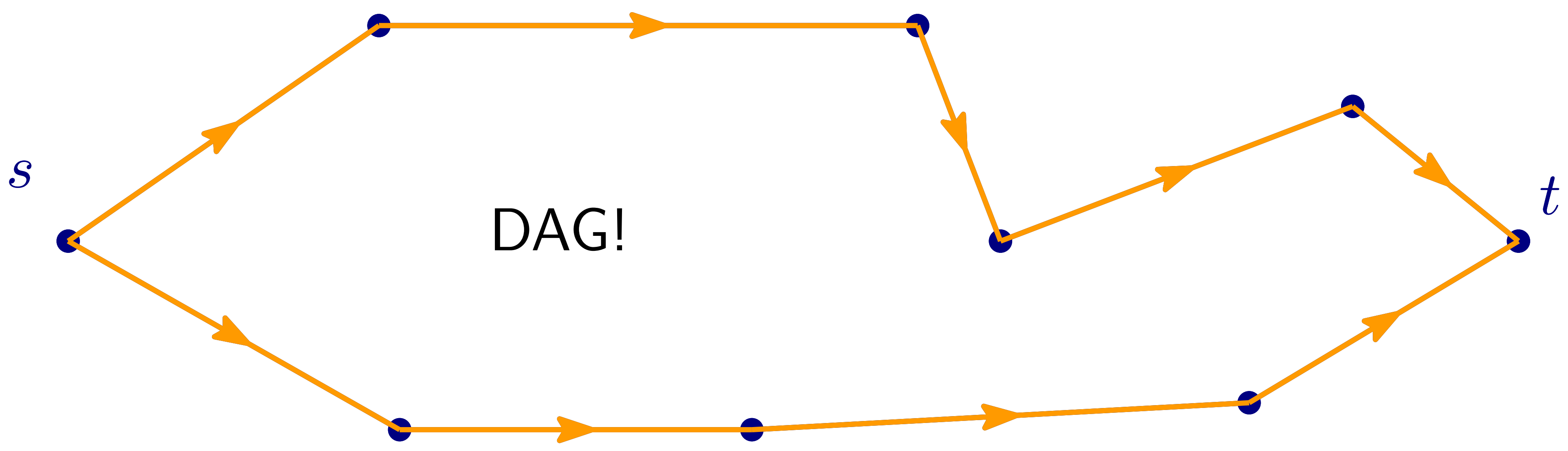


General Graphs — Attempt One

1. Compute maxflow $f^\star$
2. Look at graph induced by $f^\star$

Good edge lenghts $\boldsymbol{w}$?

- $\sum_{e \in E} \frac{n}{\boldsymbol{w}(e)}$ is small
- "Optimal" flow $f^\star$ which is short w.r.t. $\boldsymbol{w}$

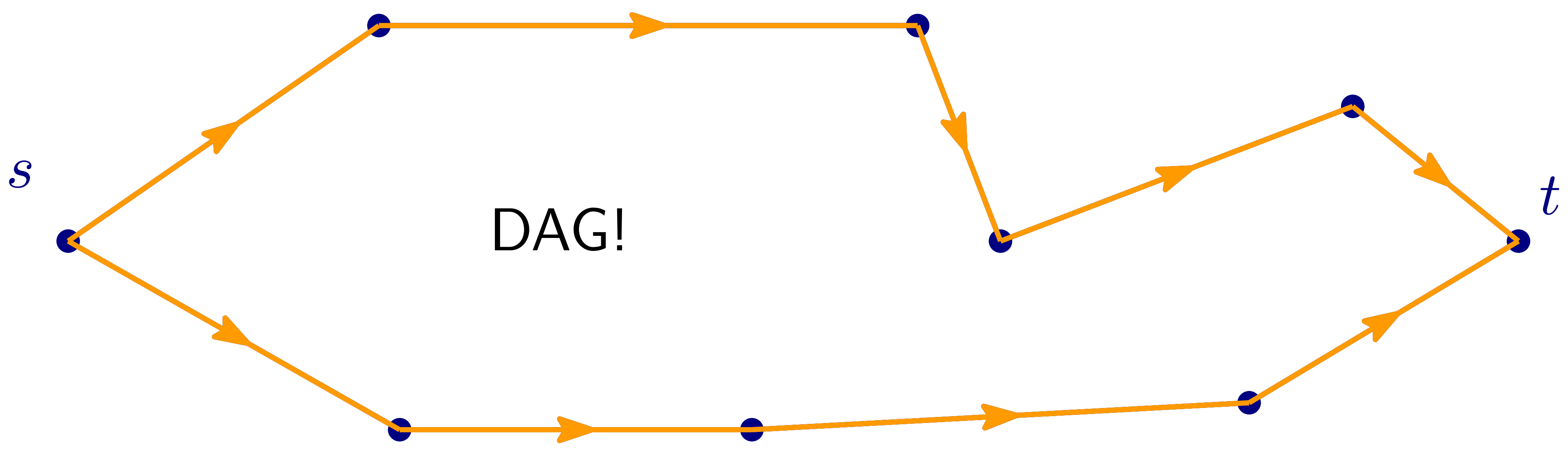


General Graphs — Attempt One

1. Compute maxflow $f^\star$
2. Look at graph induced by $f^\star$
3. Edge lengths $\boldsymbol{w}$ from topological order

Good edge lenghts $\boldsymbol{w}$?

- $\sum_{e \in E} \frac{n}{\boldsymbol{w}(e)}$ is small
- "Optimal" flow $f^\star$ which is short w.r.t. $\boldsymbol{w}$

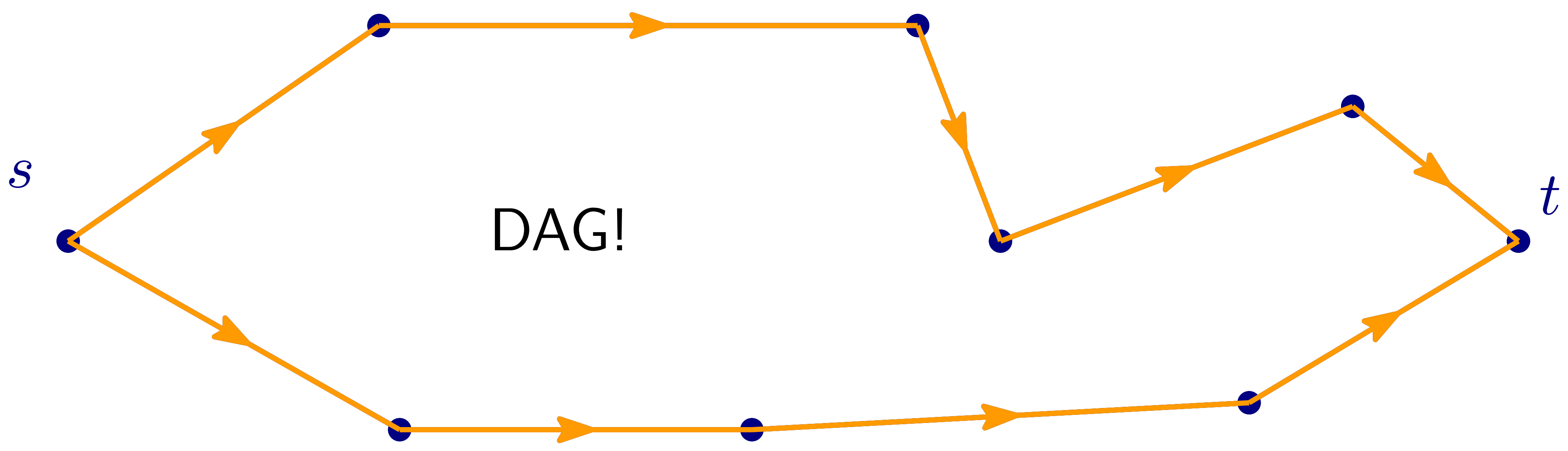


General Graphs — Attempt One

1. Compute maxflow $f^\star$
2. Look at graph induced by $f^\star$
3. Edge lengths w from topological order
4. Use weighted push-relabel to solve approx maxflow :)

Good edge lenghts w?

- $\sum_{e \in E} \frac{n}{w(e)}$ is small
- "Optimal" flow $f^\star$ which is short w.r.t. w

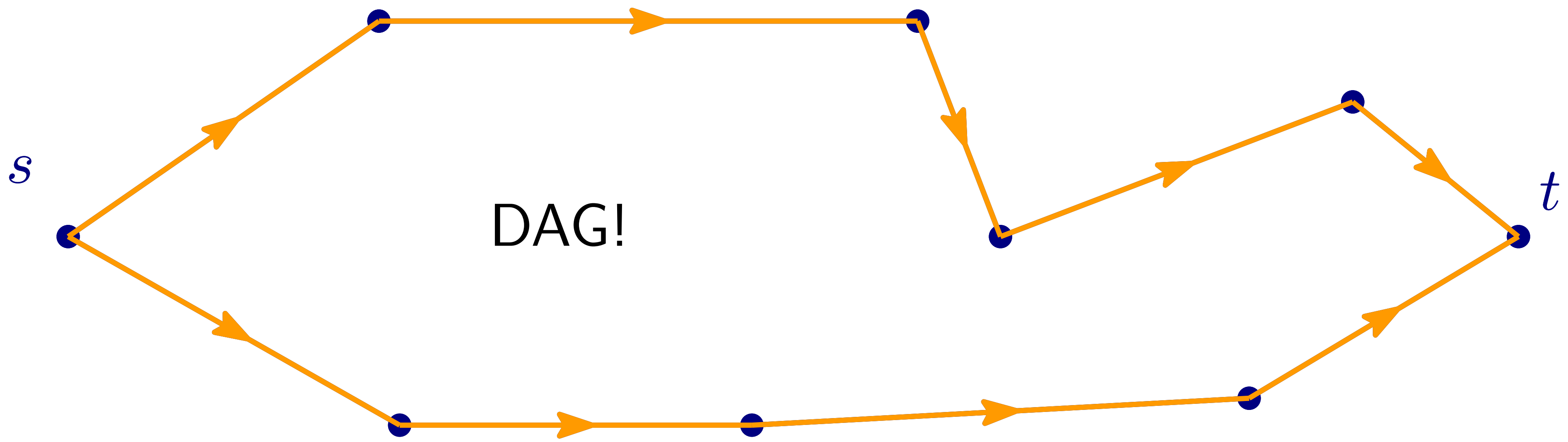


General Graphs — Attempt One

1. Compute maxflow $f^\star$ ←**Cheating!**
2. Look at graph induced by $f^\star$
3. Edge lengths $\boldsymbol{w}$ from topological order
4. Use weighted push-relabel to solve approx maxflow :)

Good edge lenghts $\boldsymbol{w}$?

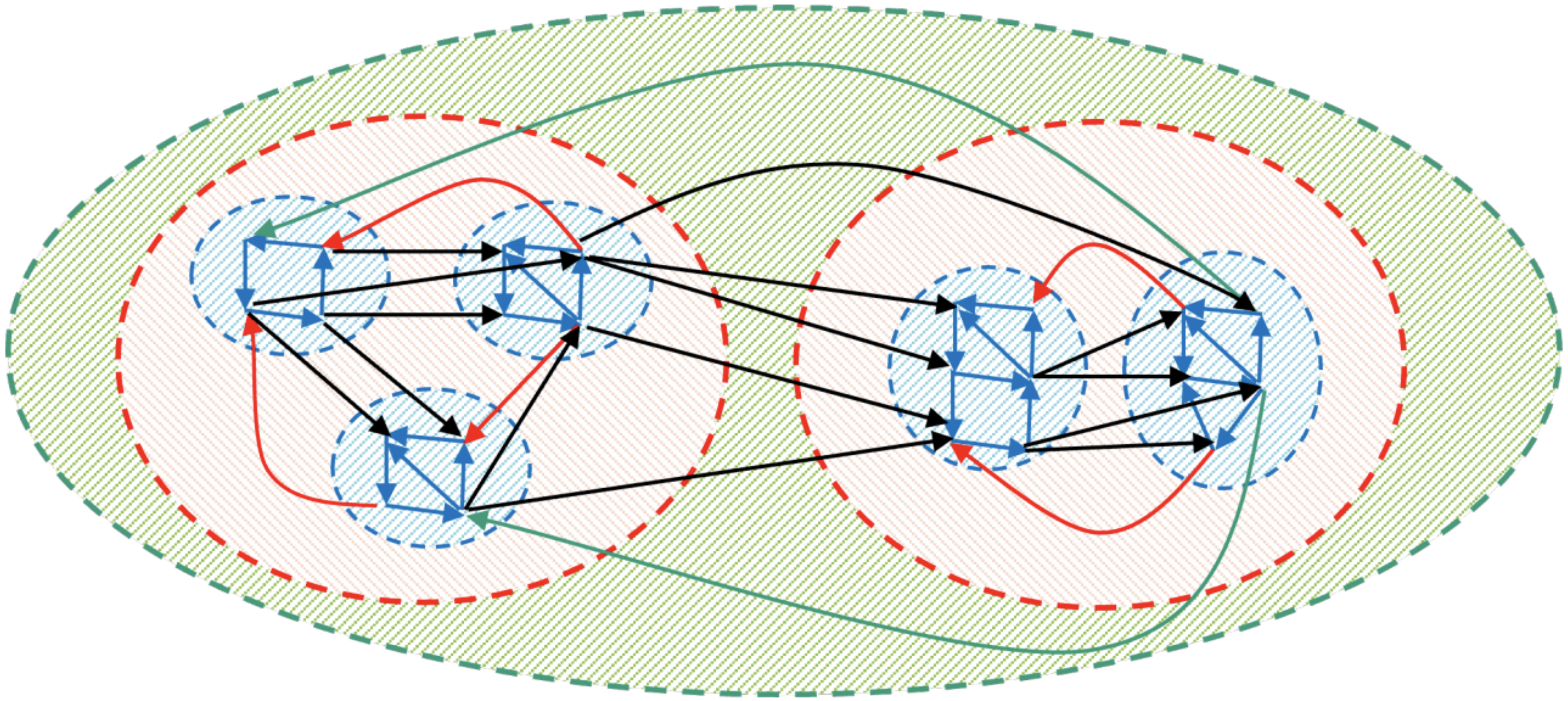
- $\sum_{e \in E} \frac{n}{\boldsymbol{w}(e)}$ is small
- "Optimal" flow $f^\star$ which is short w.r.t. $\boldsymbol{w}$



General Graphs

Good edge lenghts $\boldsymbol{w}$?

- $\sum_{e \in E} \frac{n}{\boldsymbol{w}(e)}$ is small
- "Optimal" flow $\boldsymbol{f}^{\star}$ which is short w.r.t. $\boldsymbol{w}$



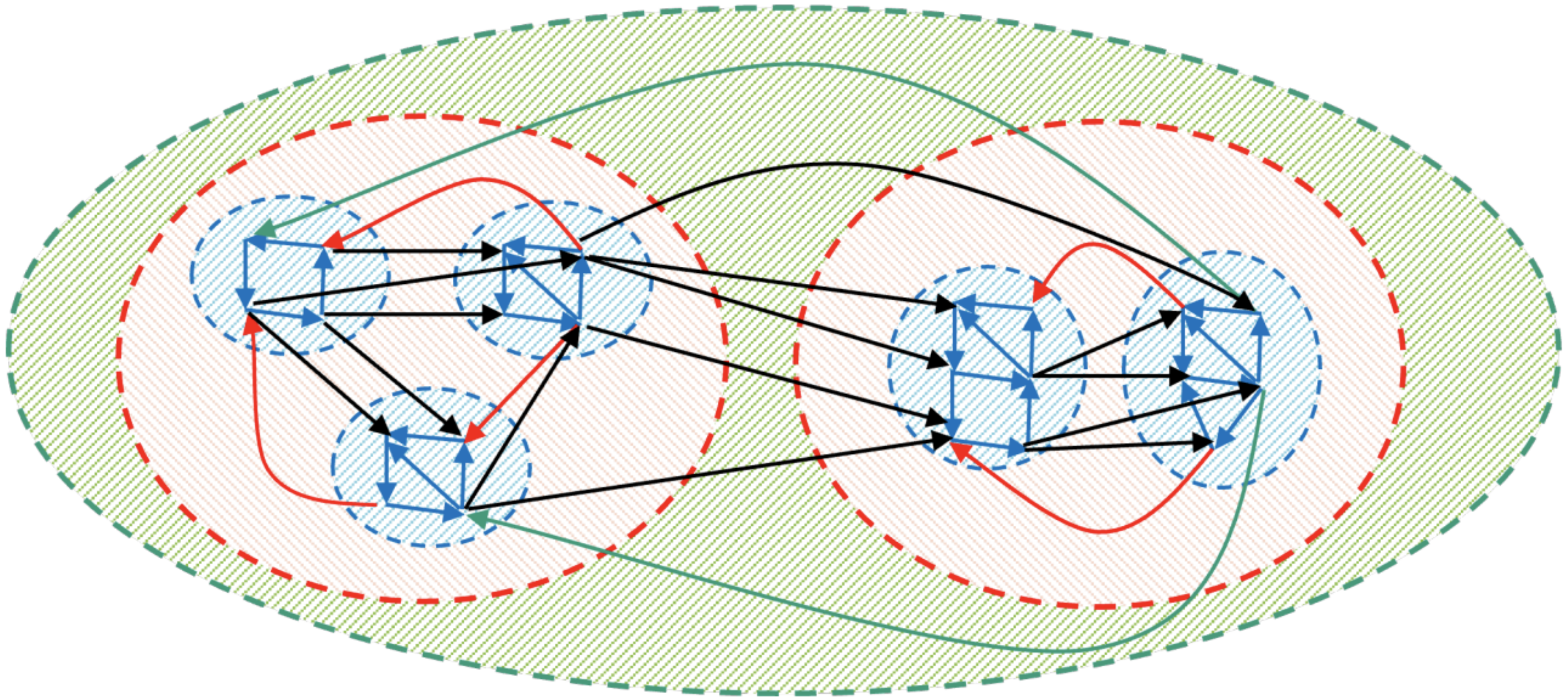
General Graphs

"Pseudo-Topological" order $\boldsymbol{\tau}$

$$\boldsymbol{w}(\boldsymbol{u},\boldsymbol{v}) = |\boldsymbol{\tau}(u) - \boldsymbol{\tau}(v)|$$

Good edge lenghts $\boldsymbol{w}$?

- ✓ $\sum_{e\in E} \frac{n}{\boldsymbol{w}(e)}$ is small
- "Optimal" flow $\boldsymbol{f}^\star$ which is short w.r.t. $\boldsymbol{w}$



General Graphs

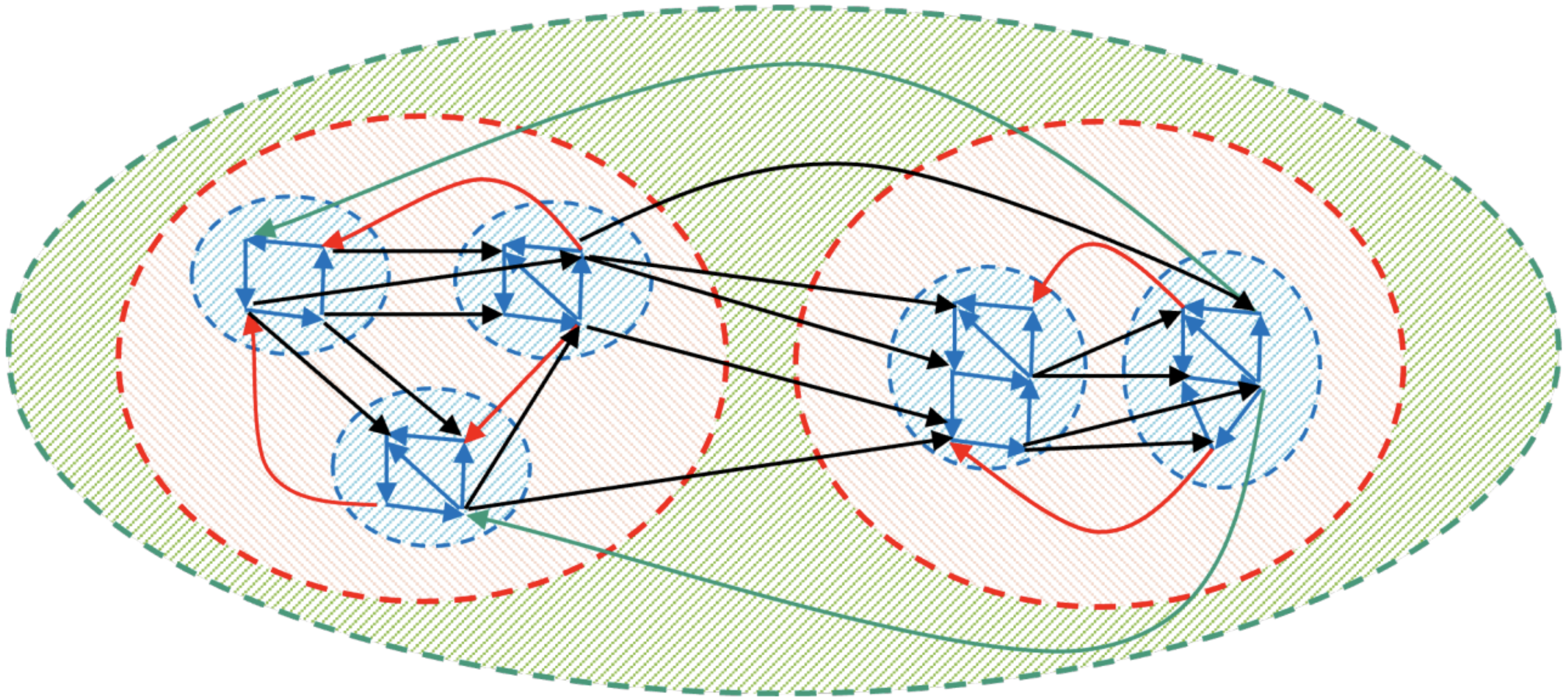
"Pseudo-Topological" order $\boldsymbol{\tau}$

$$\boldsymbol{w}(\boldsymbol{u},\boldsymbol{v}) = |\boldsymbol{\tau}(u) - \boldsymbol{\tau}(v)|$$

Directed Expander Hierarchy

Good edge lenghts $\boldsymbol{w}$?

- $\sum_{e \in E} \frac{n}{\boldsymbol{w}(e)}$ is small
- "Optimal" flow $\boldsymbol{f}^{\star}$ which is short w.r.t. $\boldsymbol{w}$



General Graphs

"Pseudo-Topological" order $\boldsymbol{\tau}$

$$\boldsymbol{w}(\boldsymbol{u},\boldsymbol{v}) = |\boldsymbol{\tau}(u) - \boldsymbol{\tau}(v)|$$

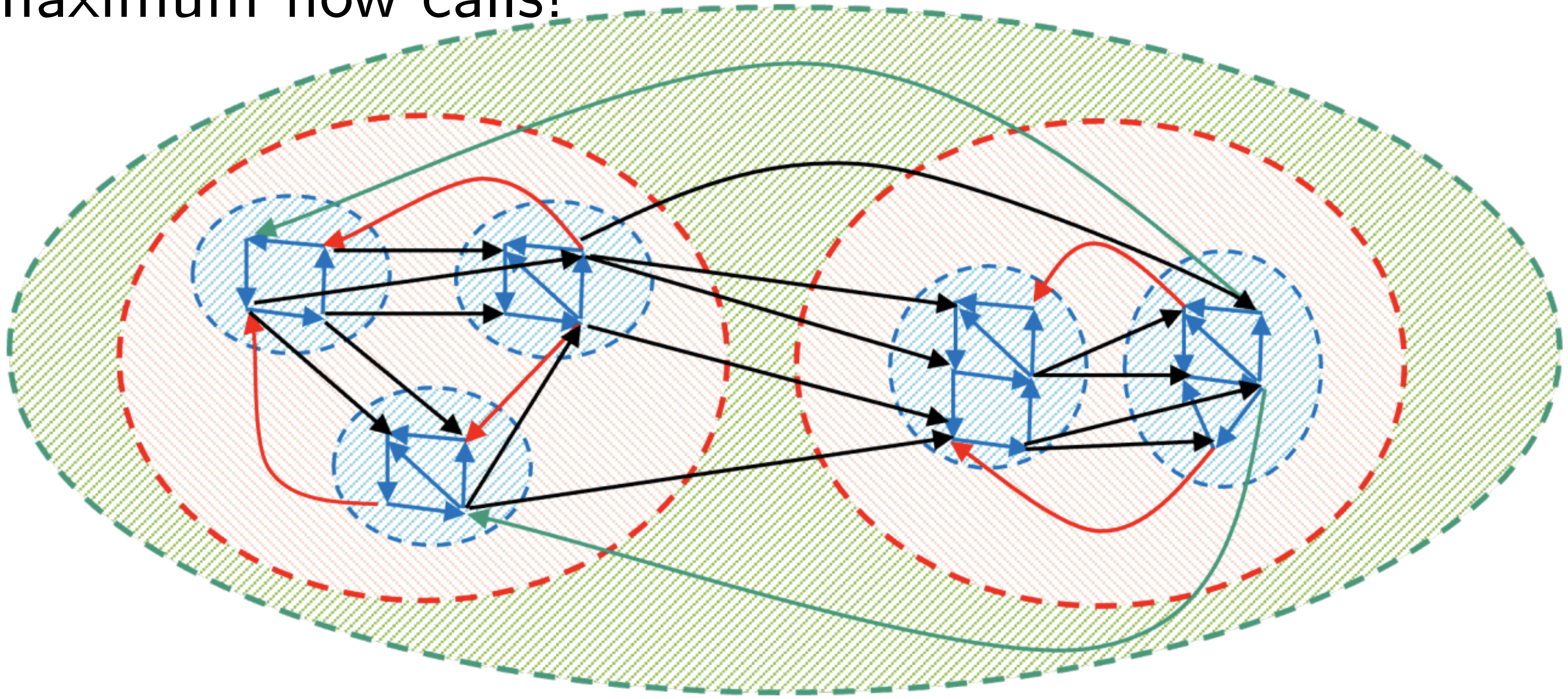
Directed Expander Hierarchy

Can build using $n^{o(1)}$ many maximum flow calls!

(Cheating!)

Good edge lenghts $\boldsymbol{w}$?

- ✓ $\sum_{e \in E} \frac{n}{\boldsymbol{w}(e)}$ is small
- "Optimal" flow $\boldsymbol{f}^{\star}$ which is short w.r.t. $\boldsymbol{w}$



General Graphs

"Pseudo-Topological" order $\boldsymbol{\tau}$

$$\boldsymbol{w}(\boldsymbol{u},\boldsymbol{v}) = |\boldsymbol{\tau}(u) - \boldsymbol{\tau}(v)|$$

Good edge lenghts $\boldsymbol{w}$?

- ✓ $\sum_{e \in E} \frac{n}{\boldsymbol{w}(e)}$ is small
- "Optimal" flow $\boldsymbol{f}^\star$ which is short w.r.t. $\boldsymbol{w}$

Directed Expander Hierarchy

Can build using $n^{o(1)}$ many maximum flow calls!

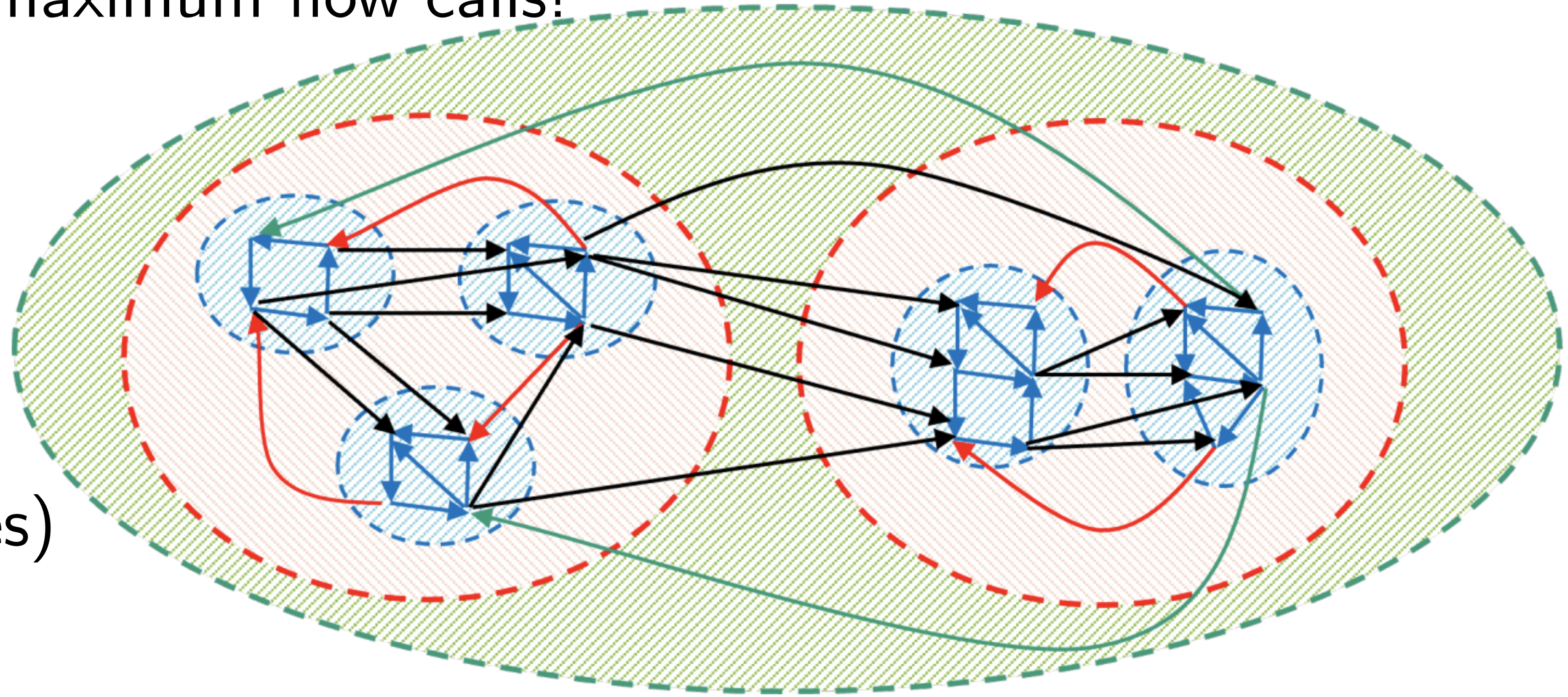
(Cheating!)

Instead:

Build Bottom Up

Bootstrap Weighted P.R.

(solve "easier" flow instances)



General Graphs

"Pseudo-Topological" order $\boldsymbol{\tau}$

$$\boldsymbol{w}(\boldsymbol{u}, \boldsymbol{v}) = |\boldsymbol{\tau}(u) - \boldsymbol{\tau}(v)|$$

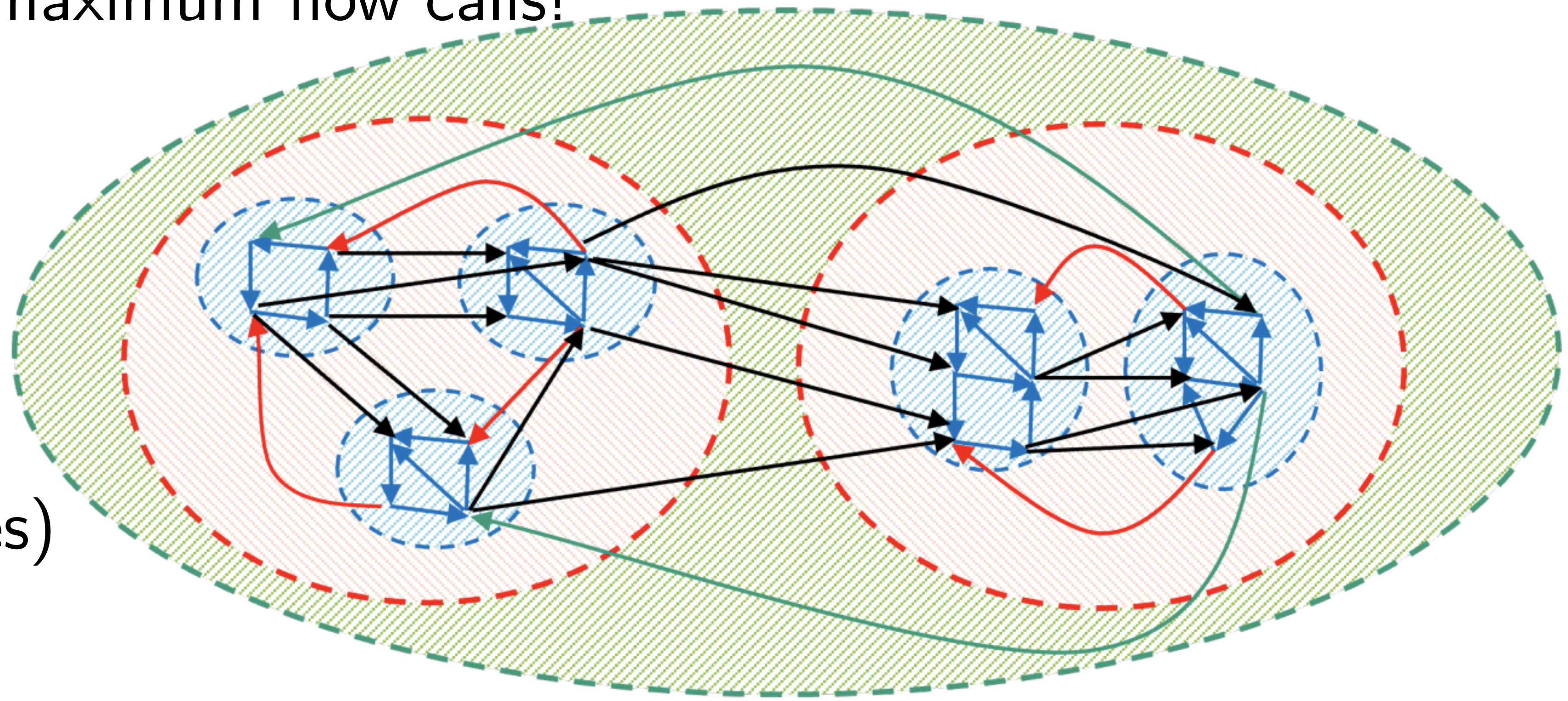
Good edge lenghts $\boldsymbol{w}$?

✓
- $\sum_{e \in E} \frac{n}{\boldsymbol{w}(e)}$ is small
- "Optimal" flow $f^{\star}$ which is short w.r.t. $\boldsymbol{w}$

Directed Expander Hierarchy

Can build using $n^{o(1)}$ many maximum flow calls!

(Cheating!)



Instead:

Build Bottom Up

Bootstrap Weighted P.R.

(solve "easier" flow instances)

Technically Complicated :(

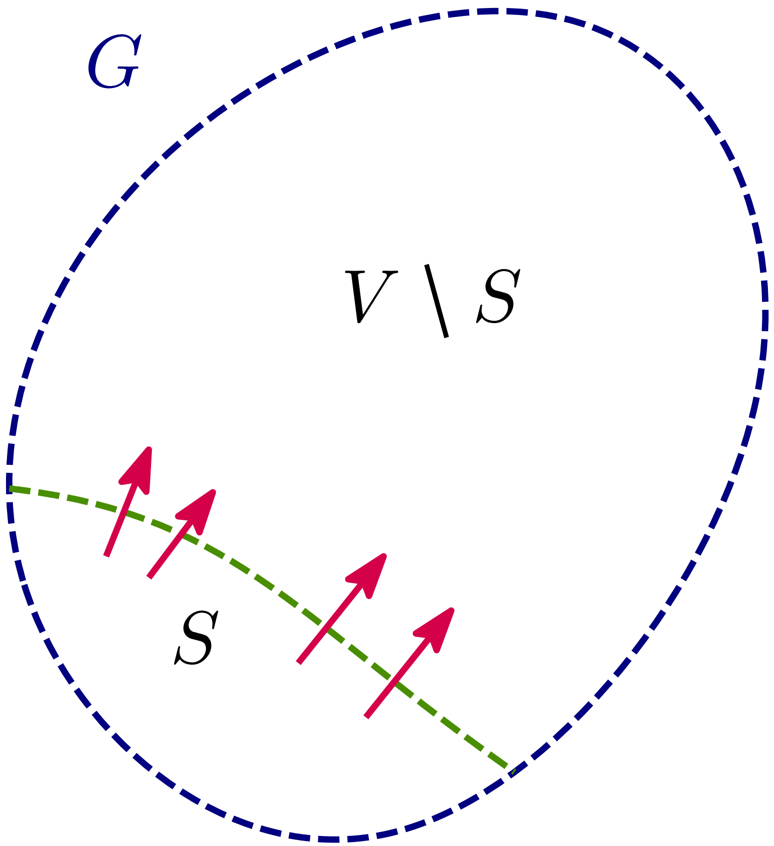
(bad guy: nestedness)

(half of our 99 page paper)

(Directed) Expanders

Def: G is ϕ-expander if $E(S, V \setminus S) \geq \phi \cdot \min\{\text{vol}(S), \text{vol}(V \setminus S)\} \quad \forall S$

$(\text{vol}(S) = \sum_{v \in S} \deg(v)$, $\phi \approx 1/n^{o(1)})$



(Directed) Expanders

Def: G is ϕ-expander if $E(S, V \setminus S) \geq \phi \cdot \min\{\text{vol}(S), \text{vol}(V \setminus S)\} \quad \forall S$

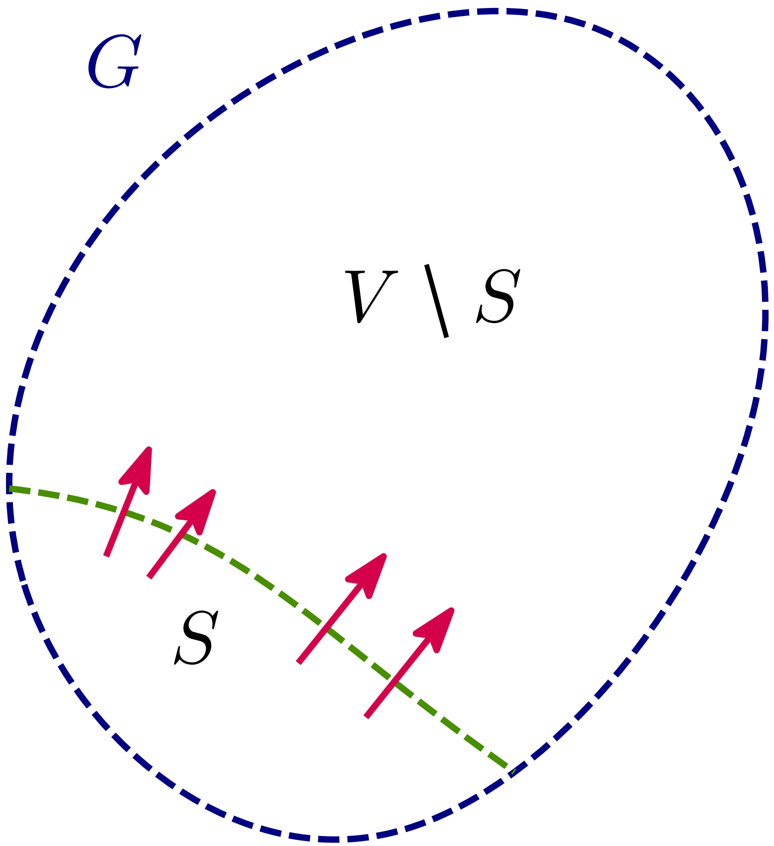
$(\text{vol}(S) = \sum_{v \in S} \deg(v)$, $\phi \approx 1/n^{o(1)})$

Examples:

Cliques

Bidirected Stars

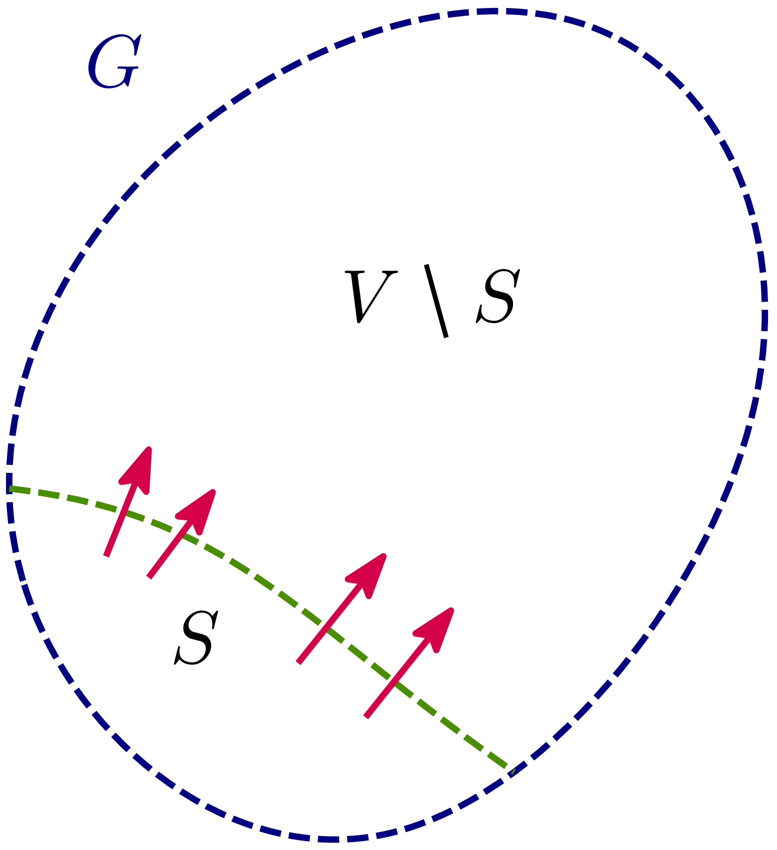
Random



(Directed) Expanders

Def: G is ϕ-expander if $E(S, V \setminus S) \geq \phi \cdot \min\{\mathrm{vol}(S), \mathrm{vol}(V \setminus S)\} \quad \forall S$

$(\mathrm{vol}(S) = \sum_{v \in S} \deg(v)$, $\phi \approx 1/n^{o(1)})$



Examples:

Cliques

Bidirected Stars

Random

Why?

Well-connected

Low diameter $\frac{\log(n)}{\phi}$

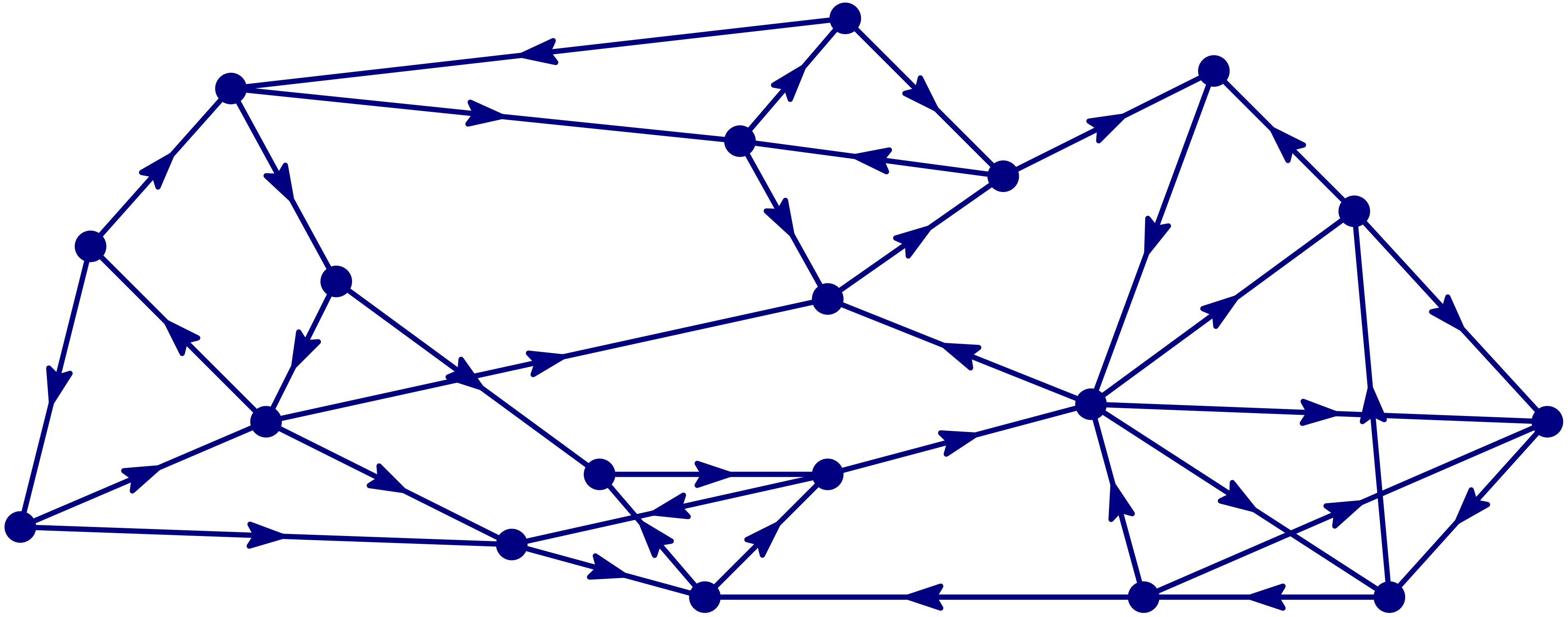
Easy to route (short) flow in

Robust to small changes

(Directed) Expander Decomposition

Every graph can be decomposed into:

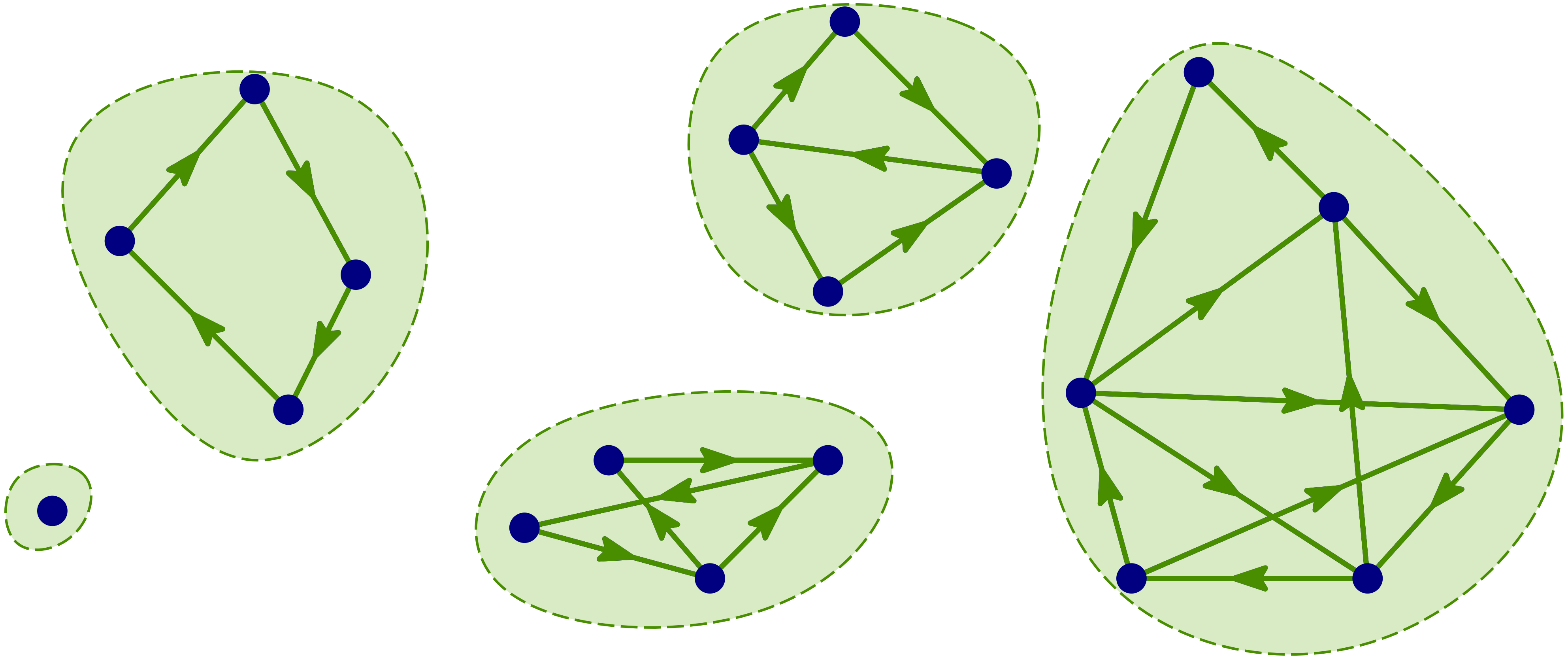
1.
2.
3.



(Directed) Expander Decomposition

Every graph can be decomposed into:

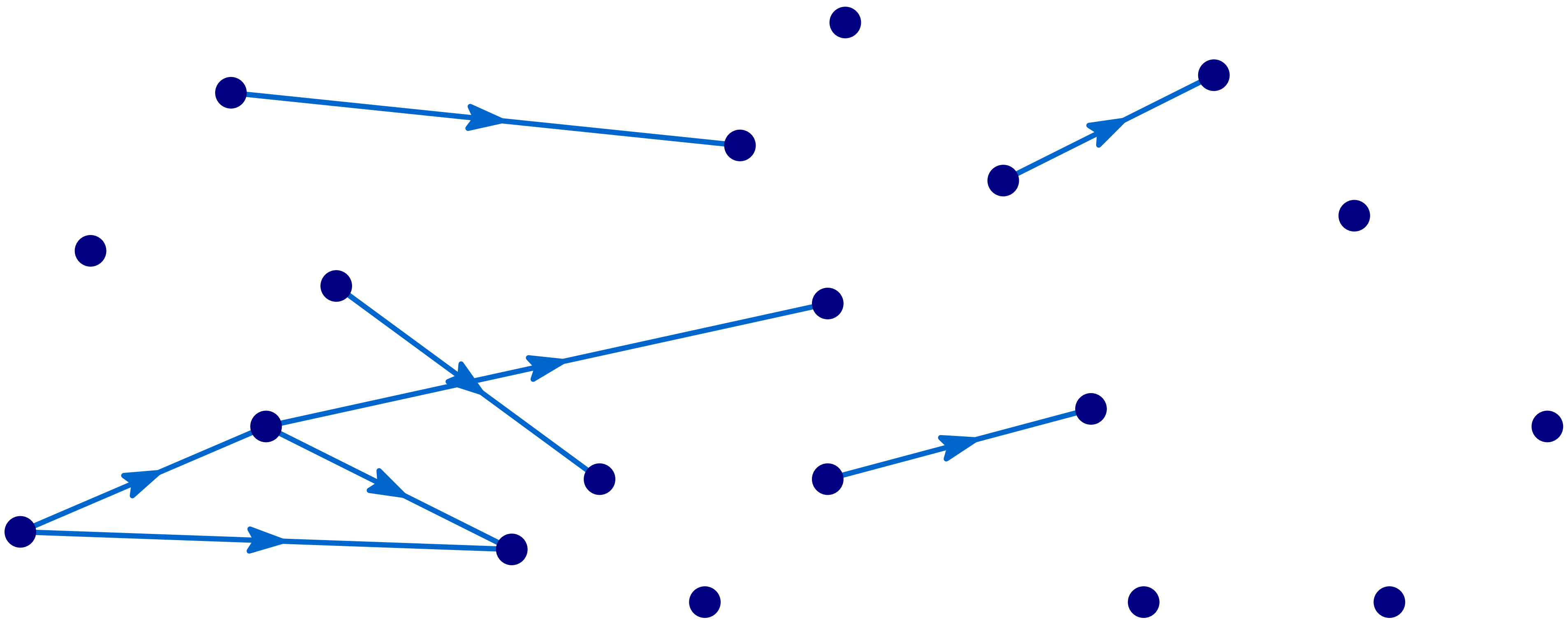
1. Expanders $X_1, \ldots, X_k$
2.
3.



(Directed) Expander Decomposition

Every graph can be decomposed into:

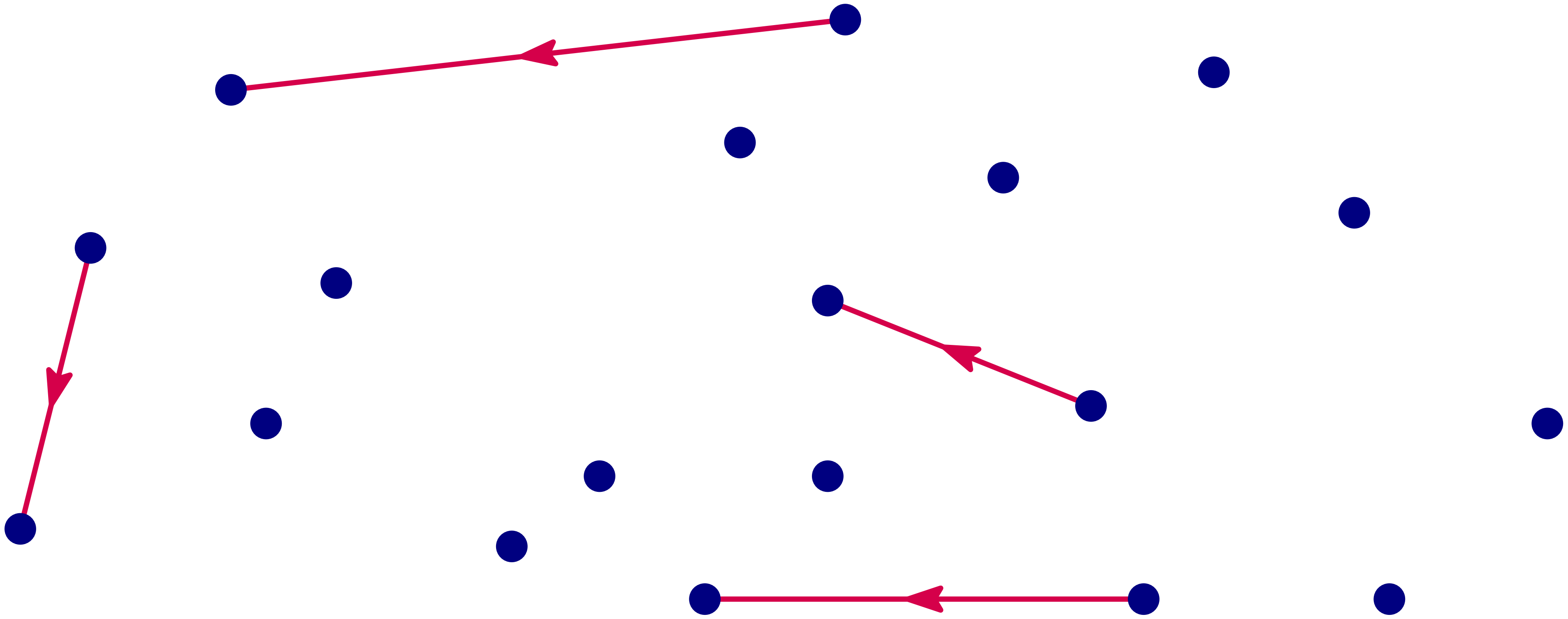
1. Expanders $X_1, \ldots, X_k$
2. DAG edges D
3.



(Directed) Expander Decomposition

Every graph can be decomposed into:

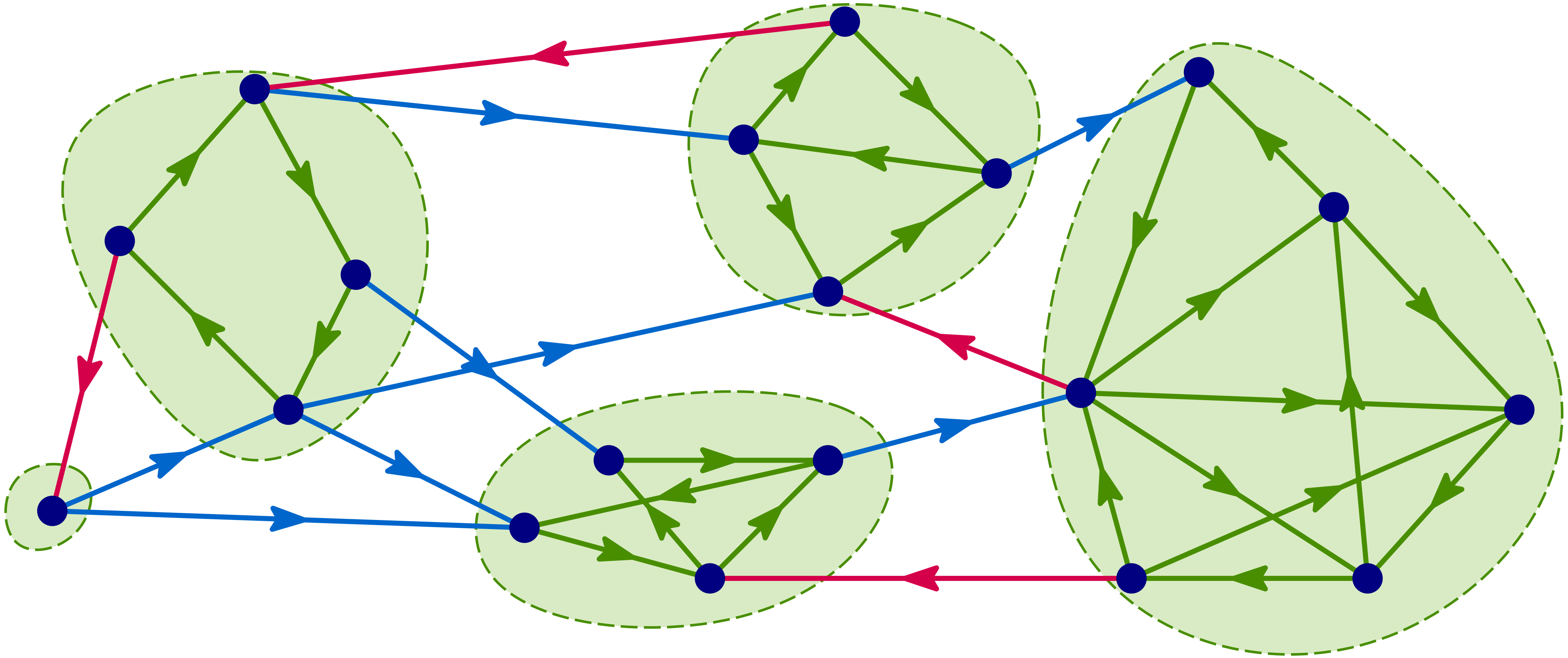
1. Expanders $X_1, \ldots, X_k$
2. DAG edges D
3. Few backward edges B



(Directed) Expander Decomposition

Every graph can be decomposed into:

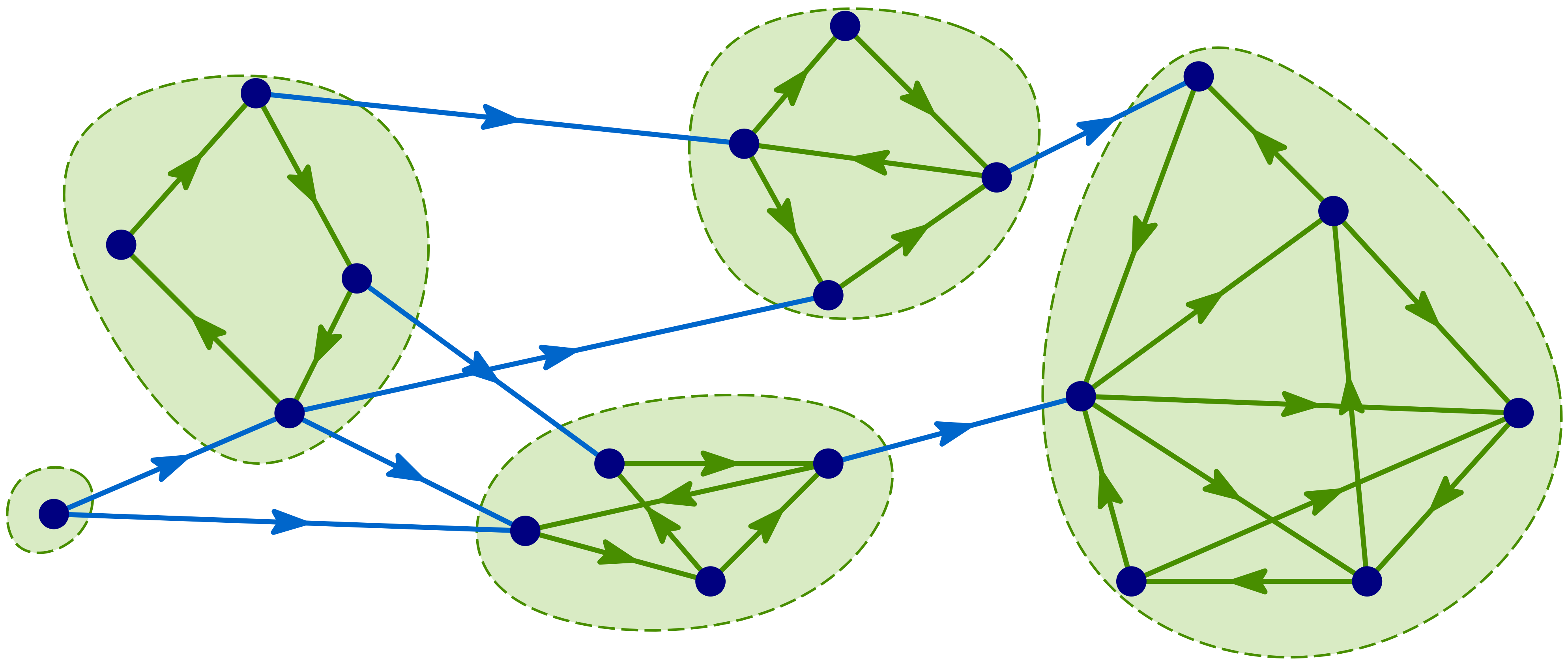
1. Expanders $X_1, \ldots, X_k$
2. DAG edges D
3. Few backward edges B



(Directed) Expander Decomposition

Every graph can be decomposed into:

1. Expanders $X_1, \ldots, X_k$ $= SCC(G \setminus B)$
2. DAG edges D
3. Few backward edges B



(Directed) Expander Decomposition

Every graph can be decomposed into:

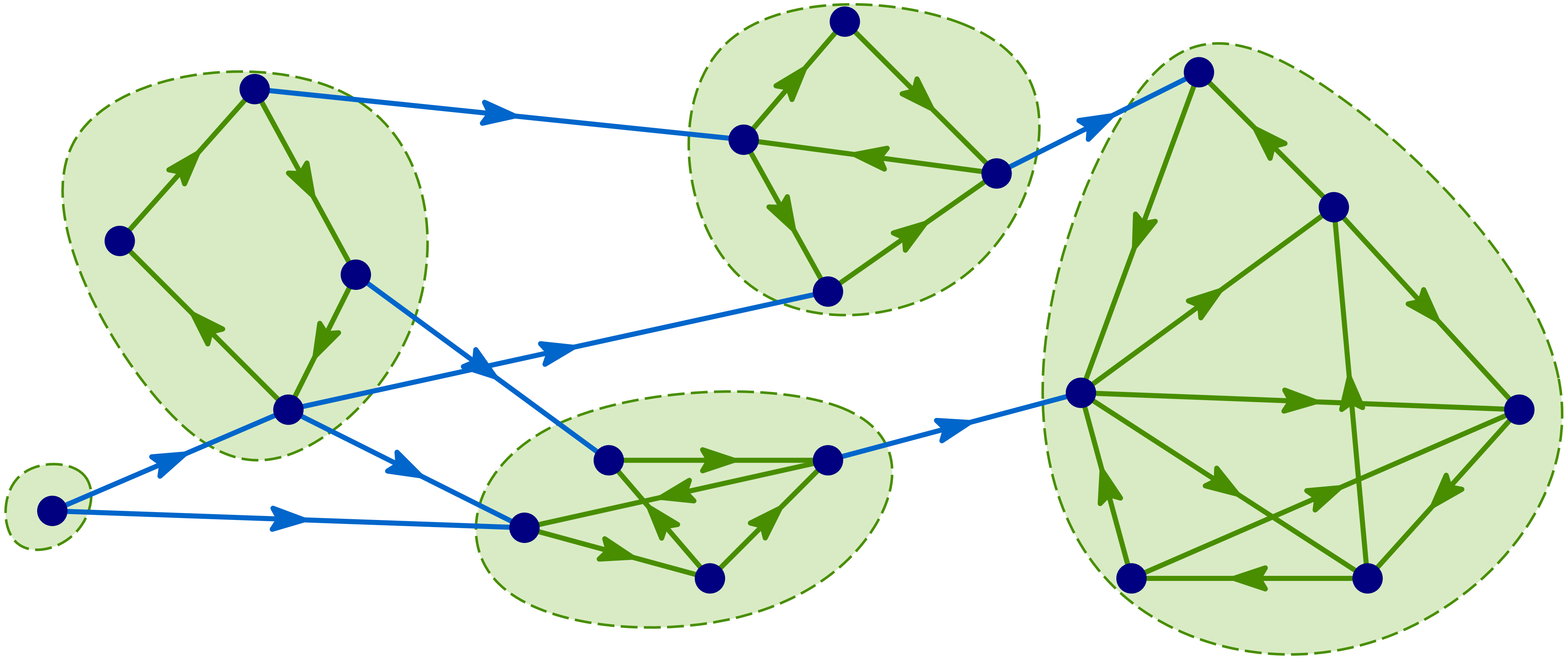
1. Expanders $X_1, \ldots, X_k$ $= SCC(G \setminus B)$
2. DAG edges D
3. Few backward edges B

Good edge lengths in $G \setminus B$:

$\boldsymbol{w}(u,v) = |\boldsymbol{\tau}(u) - \boldsymbol{\tau}(v)|$

$\boldsymbol{\tau}$ respects DAG

$\boldsymbol{\tau}$ contiguous in expanders



(Directed) Expander Decomposition

Every graph can be decomposed into:

1. Expanders $X_1, \ldots, X_k$ $= SCC(G \setminus B)$
2. DAG edges D
3. Few backward edges B

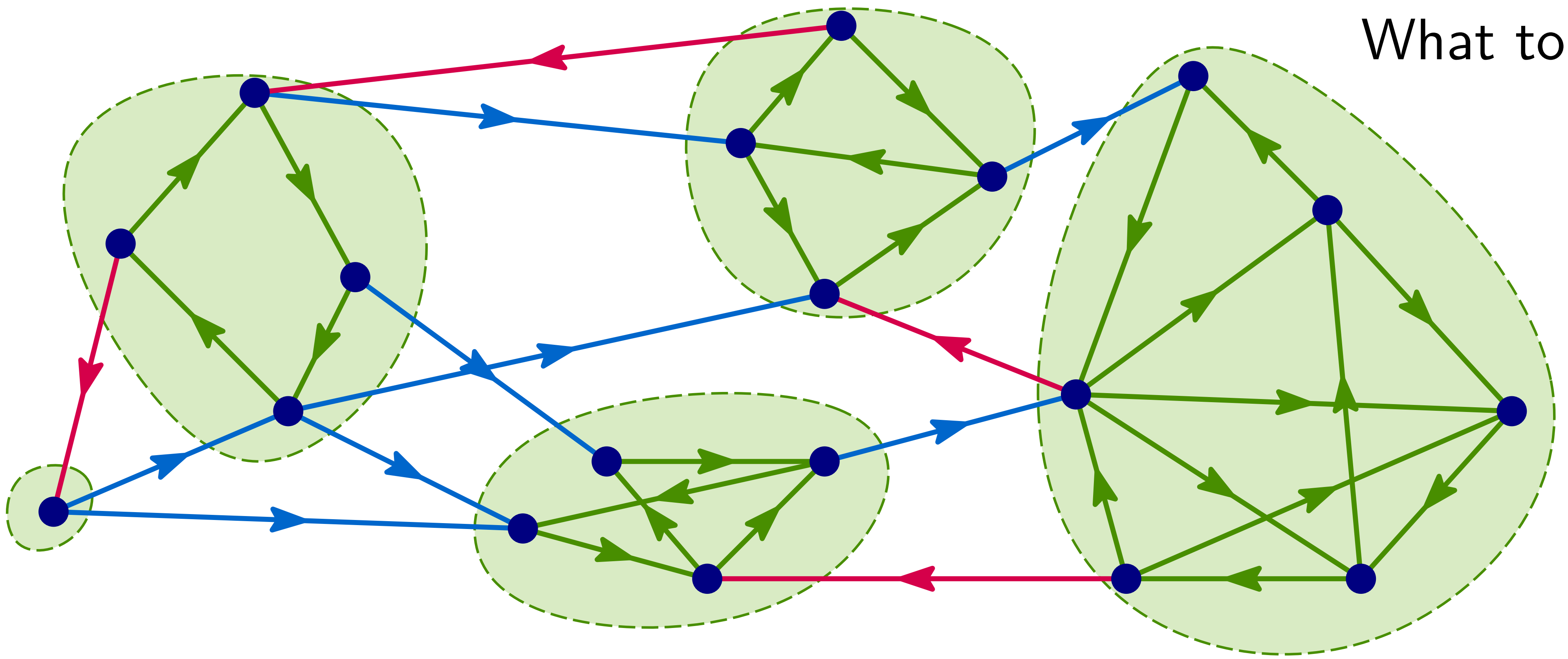
Good edge lengths in $G \setminus B$:

$$\boldsymbol{w}(u,v) = |\boldsymbol{\tau}(u) - \boldsymbol{\tau}(v)|$$

$\boldsymbol{\tau}$ respects DAG

$\boldsymbol{\tau}$ contiguous in expanders

What to do about B?



(Directed) Expander Decomposition

Every graph can be decomposed into:

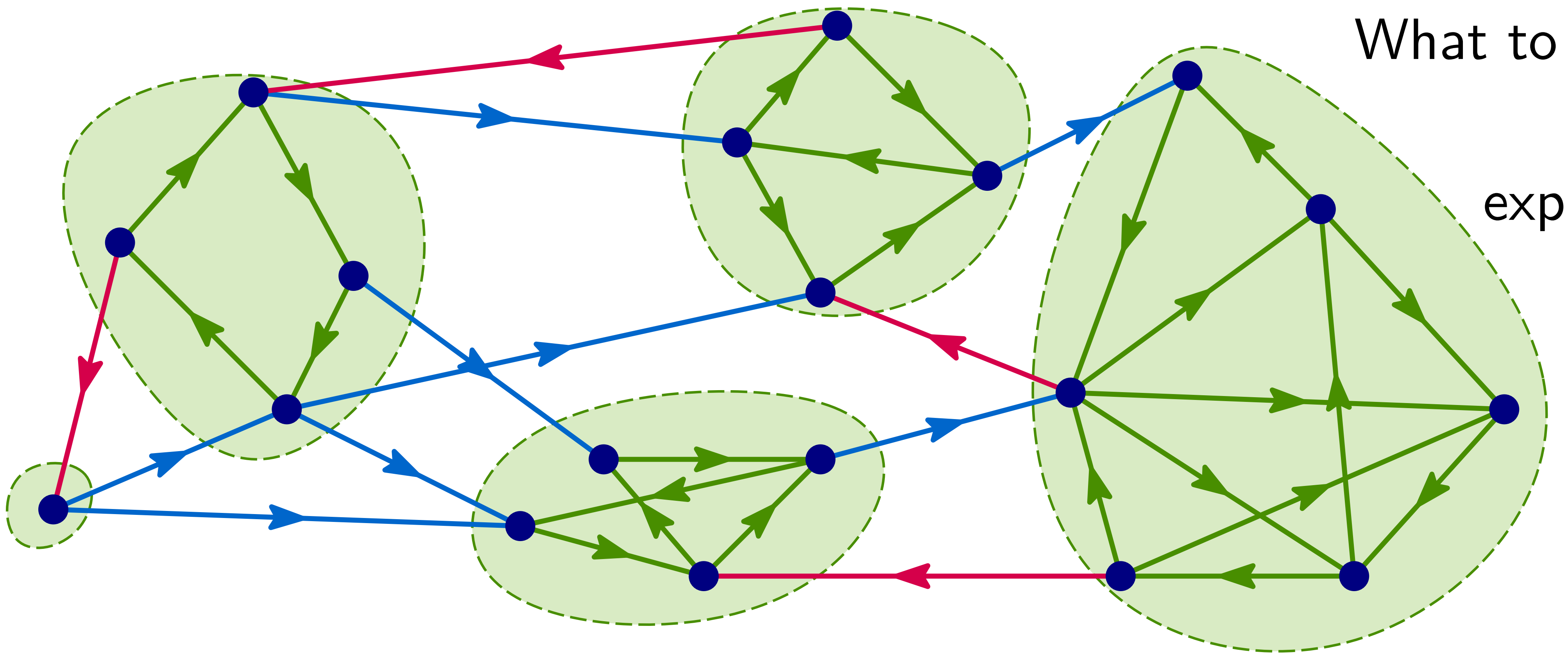
1. Expanders $X_1, \dots, X_k$ = $SCC(G \setminus B)$
2. DAG edges D
3. Few backward edges B

Good edge lengths in $G \setminus B$:

$\boldsymbol{w}(u, v) = |\boldsymbol{\tau}(u) - \boldsymbol{\tau}(v)|$

$\boldsymbol{\tau}$ respects DAG

$\boldsymbol{\tau}$ contiguous in expanders



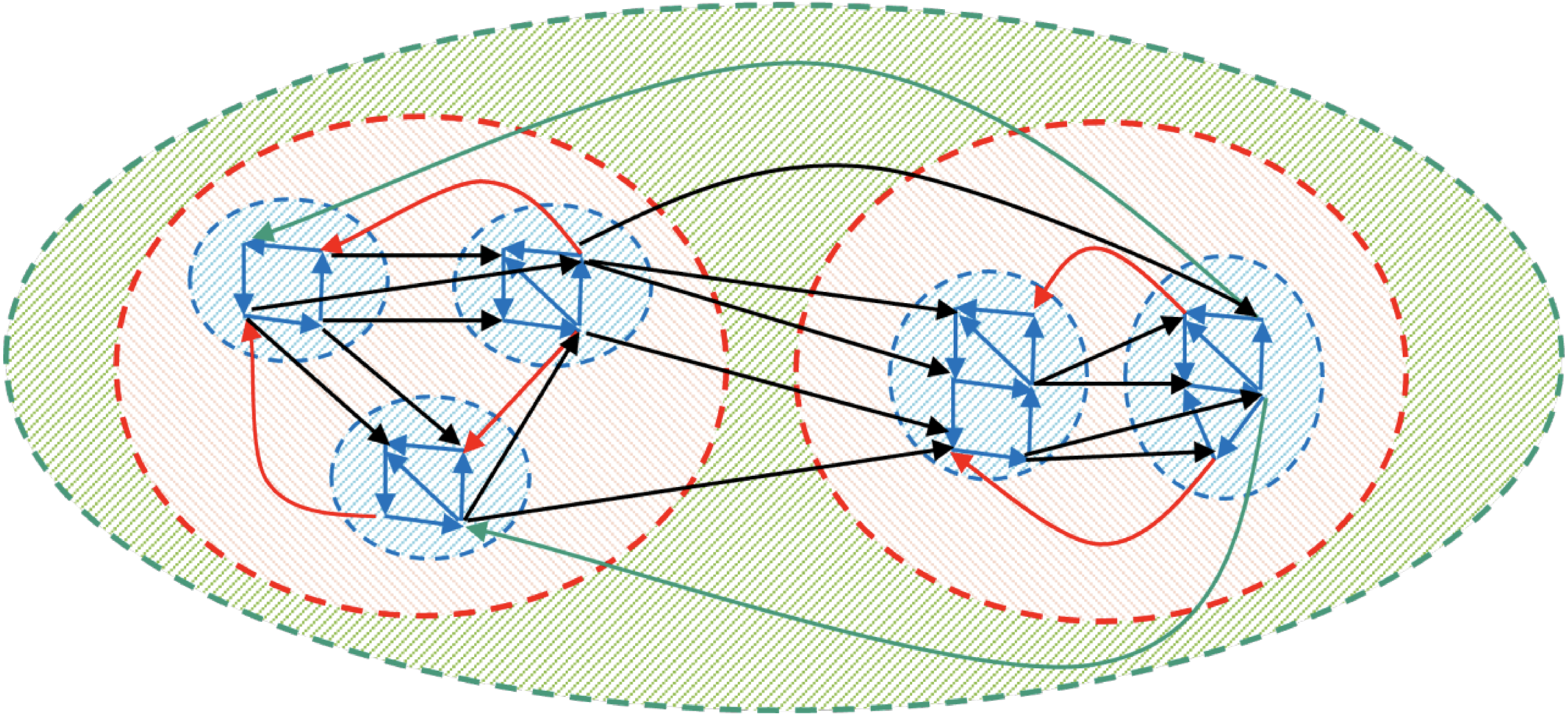
What to do about B?

recurse!

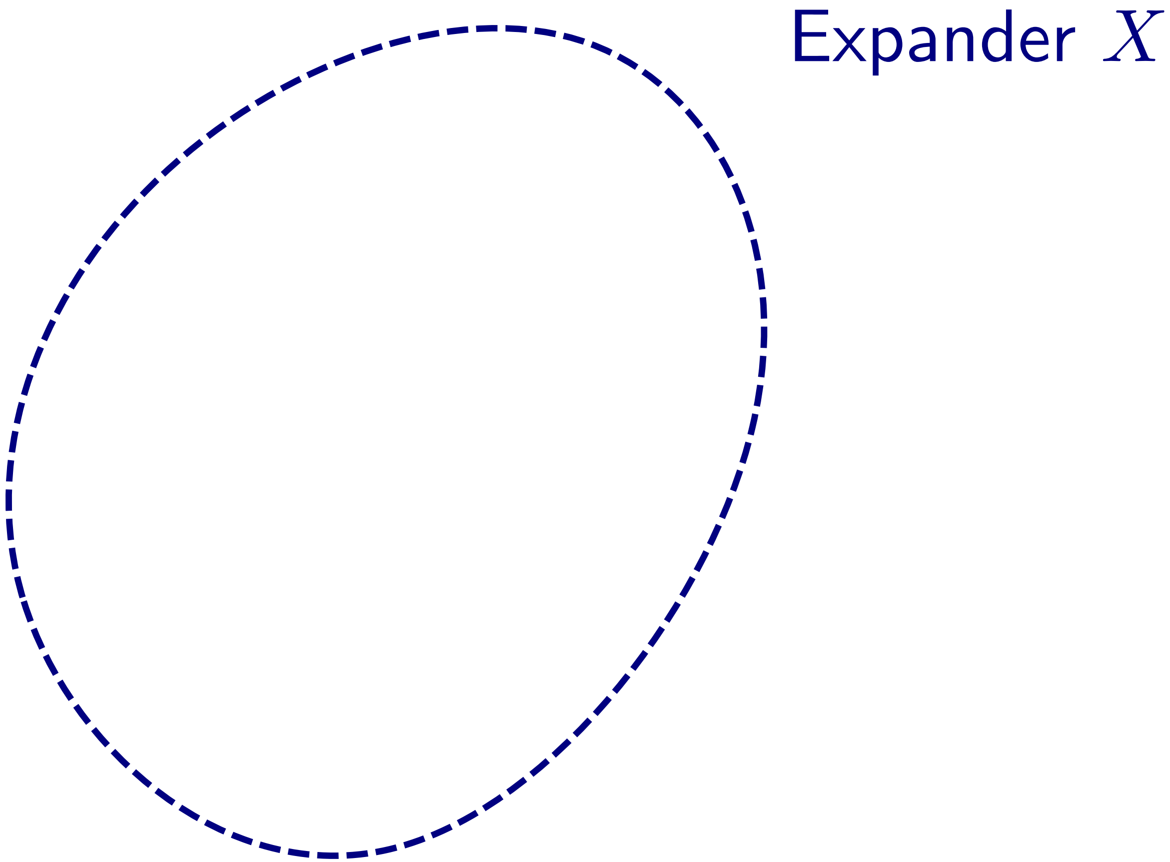
expander decomp.

w.r.t B

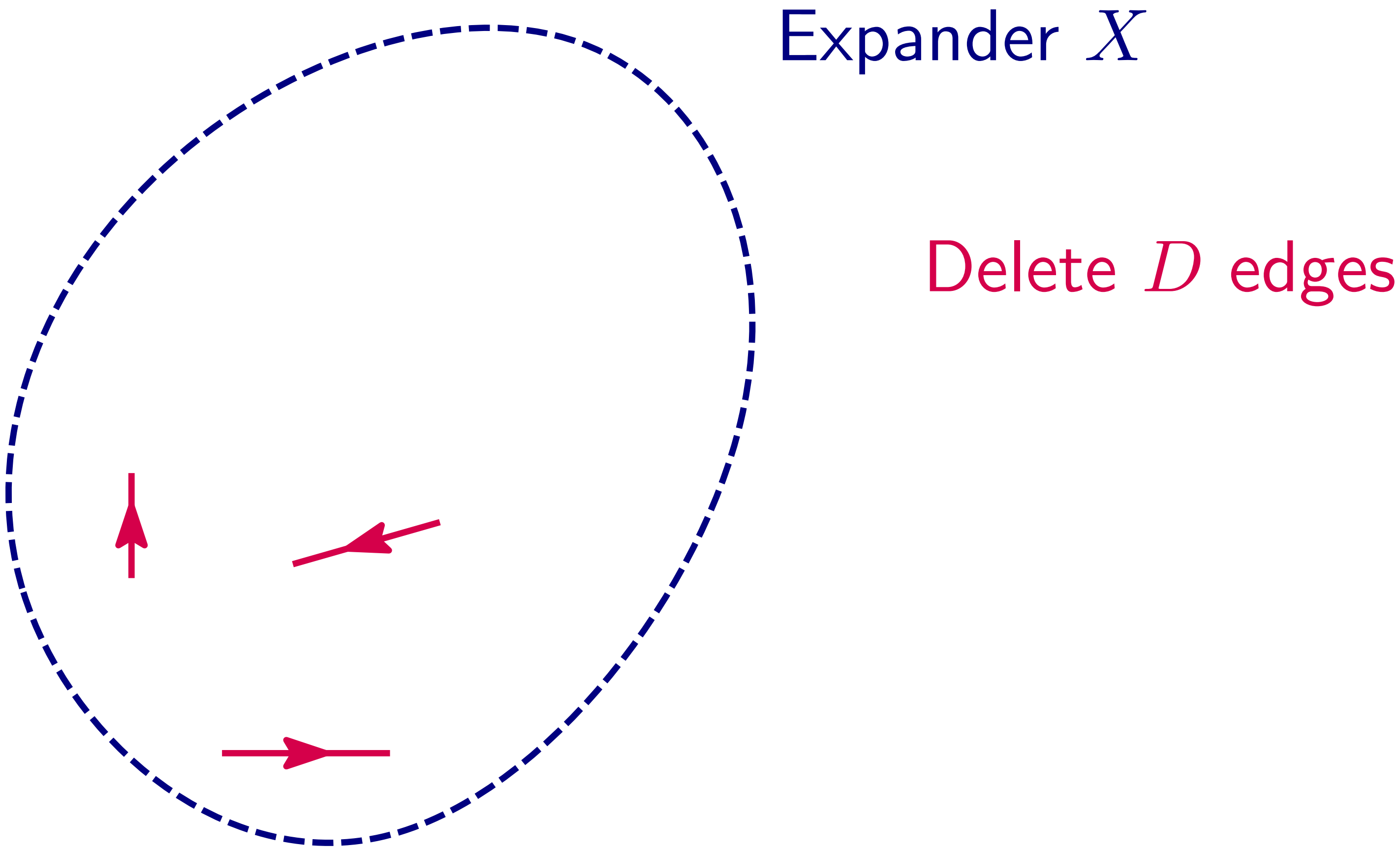
(Directed) Expander Decomposition



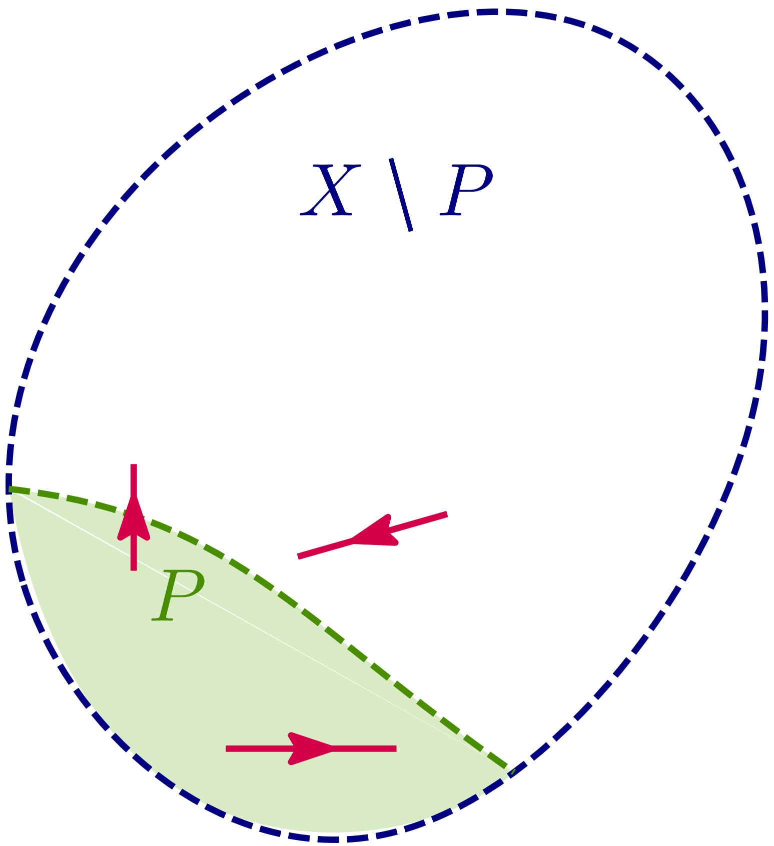
Technique Highlight: Path-Reversal Expander Pruning



Technique Highlight: Path-Reversal Expander Pruning



Technique Highlight: Path-Reversal Expander Pruning



Expander X

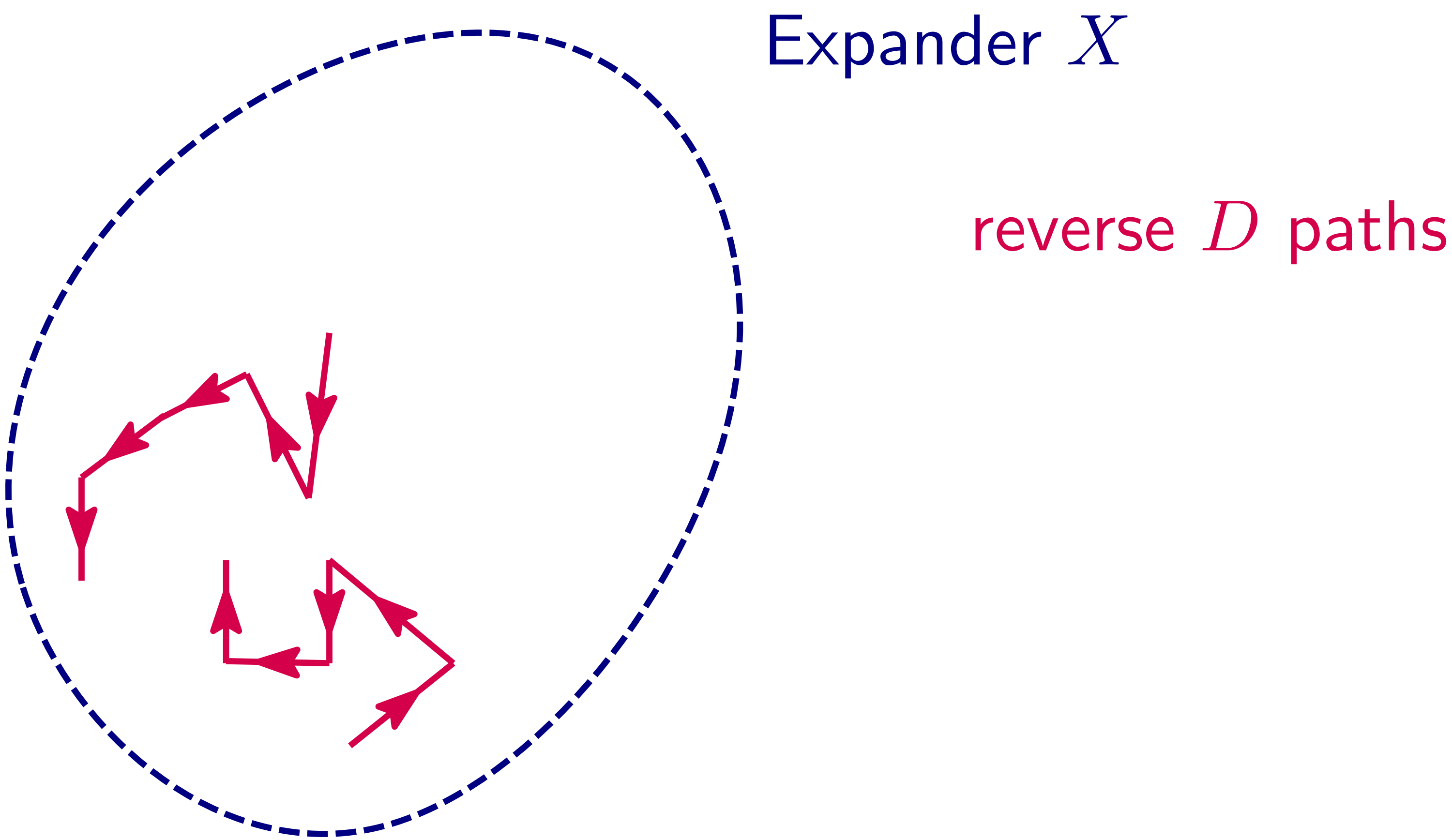
Delete D edges

Small “pruned” part P $\mathrm{vol}(P) \leq 6|D|/\phi$

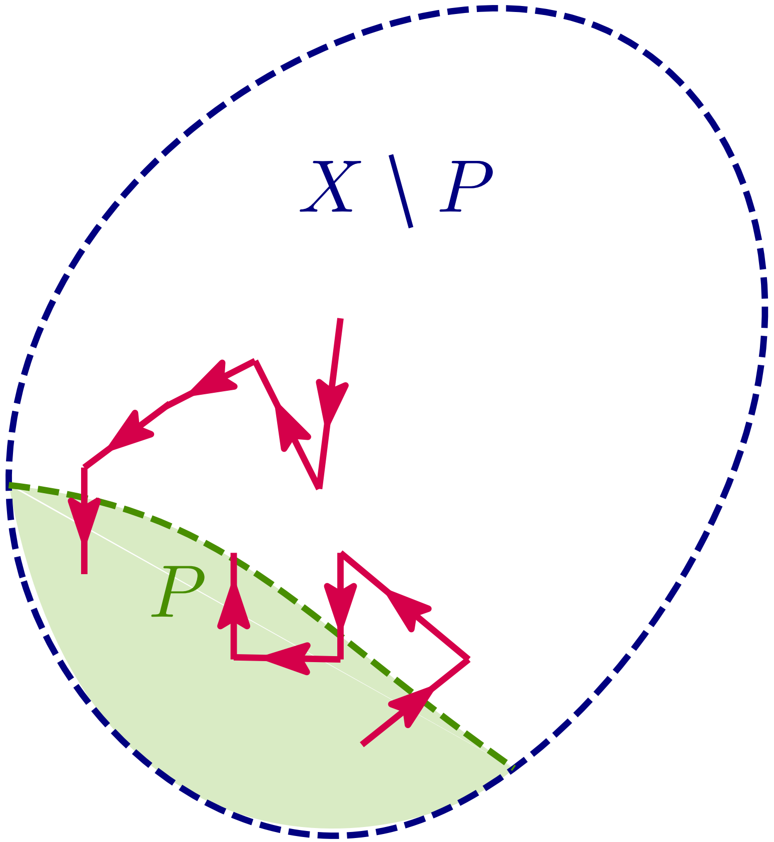
$X \setminus P$ is still expander

Known: “Expander Pruning”

Technique Highlight: Path-Reversal Expander Pruning



Technique Highlight: Path-Reversal Expander Pruning



Expander X

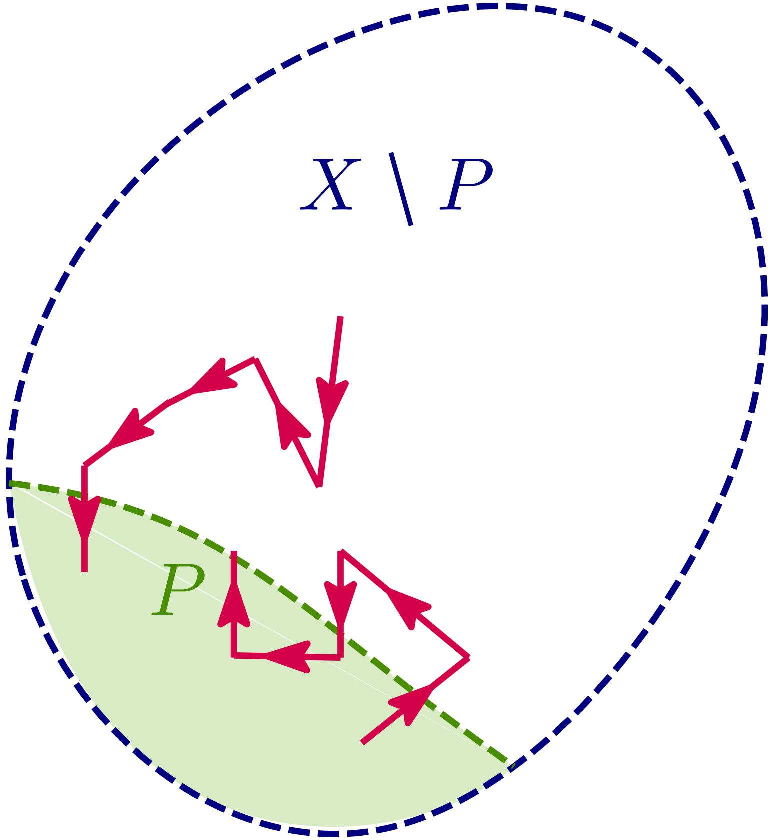
reverse D paths

Small “pruned” part P $\mathrm{vol}(P) \leq 6|D|/\phi$

$X \setminus P$ is still expander

Theorem: Path-Reversal Expander Pruning

Technique Highlight: Path-Reversal Expander Pruning



Expander X

reverse D paths

Small “pruned” part P $\quad \mathrm{vol}(P) \leq 6|D|/\phi$

$X \setminus P$ is still expander

Theorem: Path-Reversal Expander Pruning

Directed Expander Hierarchy is robust under flow augmentation

Bottleneck towards $\tilde{O}(m)$:
Approximate Max Flow in DAGs

Summary & Open Problems

Summary

Main Result: Maximum flow in on n-vertex graphs in $n^{2+o(1)}$ time.

Summary

Main Result: Maximum flow in on n-vertex graphs in $n^{2+o(1)}$ time.

First combinatorial / augmenting-path improvement since $O(m \cdot \min\{\sqrt{m}, n^{2/3}\})$

[Karzanov'73]

[Even-Tarjan'75]

[Goldberg-Rao'98]

Summary

Main Result: Maximum flow in on n-vertex graphs in $n^{2+o(1)}$ time.

First combinatorial / augmenting-path improvement since $O(m \cdot \min\{\sqrt{m}, n^{2/3}\})$

[Karzanov'73]

[Even-Tarjan'75]

[Goldberg-Rao'98]

Techniques:

- Augmenting Paths (new version of Push-Relabel)
- Directed Expander Hierarchy
- Mostly Self-Contained

Summary

Main Result: Maximum flow in on n-vertex graphs in $n^{2+o(1)}$ time.

First combinatorial / augmenting-path improvement since $O(m \cdot \min\{\sqrt{m}, n^{2/3}\})$

[Karzanov'73]

[Even-Tarjan'75]

[Goldberg-Rao'98]

Techniques:

Augmenting Paths (new version of Push-Relabel)

Directed Expander Hierarchy

Mostly Self-Contained

Open Questions:

"Simple", "Combinatorial", "Implementable": $E^{1+o(1)}$ or $\tilde{O}(E)$ Maximum Flow?

(bottleneck: apx. maxflow on DAG)

Minimum Cost Maximum Flow, General Matching, Matroid Intersection, ...

Summary

Main Result: Maximum flow in on n-vertex graphs in $n^{2+o(1)}$ time.

First combinatorial / augmenting-path improvement since $O(m \cdot \min\{\sqrt{m}, n^{2/3}\})$

[Karzanov'73]

[Even-Tarjan'75]

[Goldberg-Rao'98]

Techniques:

Augmenting Paths (new version of Push-Relabel)

Directed Expander Hierarchy

Mostly Self-Contained

Thanks!

Open Questions:

"Simple", "Combinatorial", "Implementable": $E^{1+o(1)}$ or $\tilde{O}(E)$ Maximum Flow?

(bottleneck: apx. maxflow on DAG)

Minimum Cost Maximum Flow, General Matching, Matroid Intersection, ...

Comparision

Ours

[Chen-Kyng-Liu-Peng-Gutenberg-Sachdeva'22]

Comparision

Ours	[Chen-Kyng-Liu-Peng-Gutenberg-Sachdeva'22]
Maximum Flow	Minimum Cost Maximum Flow

Comparision

Ours

Maximum Flow

$n^{2+o(1)}$

[Chen-Kyng-Liu-Peng-Gutenberg-Sachdeva'22]

Minimum Cost Maximum Flow

$m^{1+o(1)}$

Comparision

Ours	[Chen-Kyng-Liu-Peng-Gutenberg-Sachdeva'22]
Maximum Flow	Minimum Cost Maximum Flow
$n^{2+o(1)}$	$m^{1+o(1)}$
Combinatorial Augmenting Paths	Continuous Optimization Dynamic Data Structures
Implementable?	Tricky to implement