

Combinatorial Maximum Flow via Weighted Push-Relabel on Shortcut Graphs

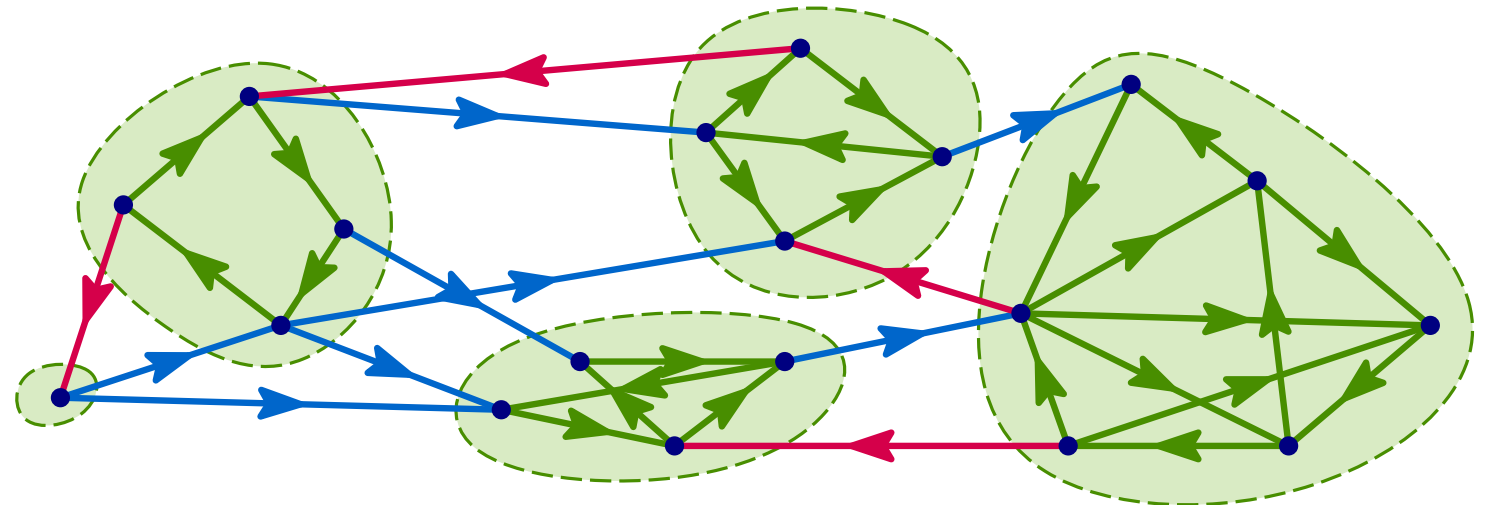
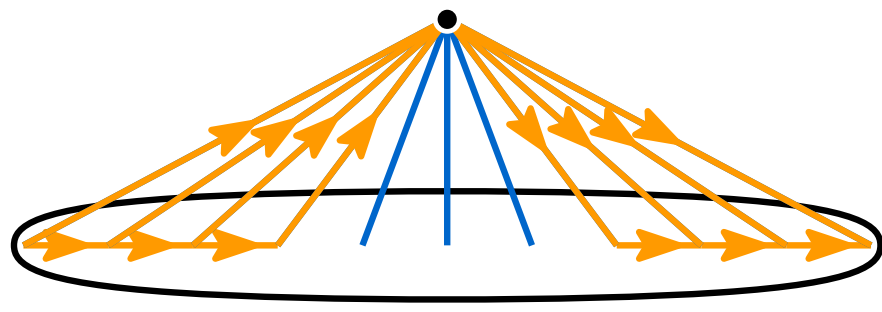
Aaron Bernstein

Joakim Blikstad

Jason Li

Thatchaphol Saranurak

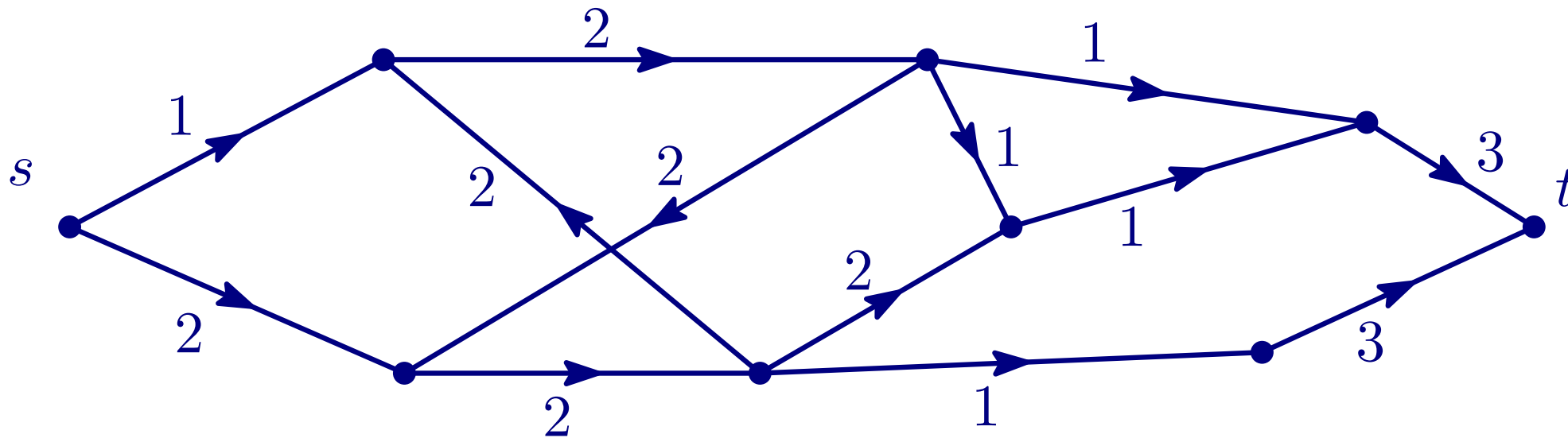
Ta-Wei Tu



Maximum Flow

Given: Directed graph $G = (V, E)$, edge capacities $c : E \rightarrow \mathbb{Z}_{\geq 1}$, source s , and sink t .

Goal: Compute s, t -flow f of largest size.

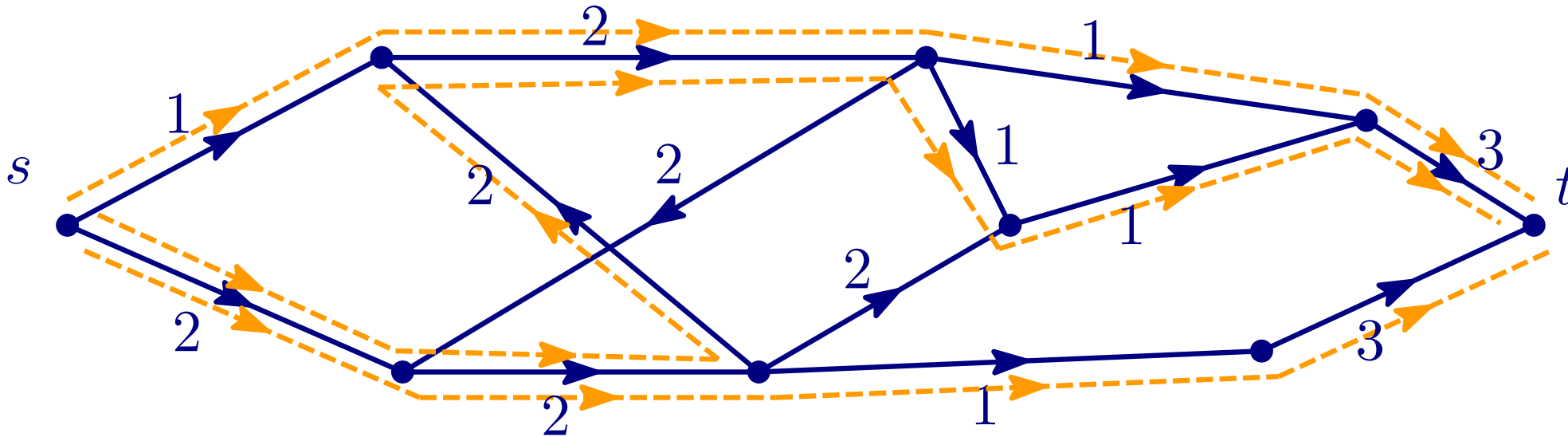


Maximum Flow

Given: Directed graph $G = (V, E)$, edge capacities $c : E \rightarrow \mathbb{Z}_{\geq 1}$, source s , and sink t .

Goal: Compute s, t -flow f of largest size.

$$|f| = 3$$



Maximum Flow

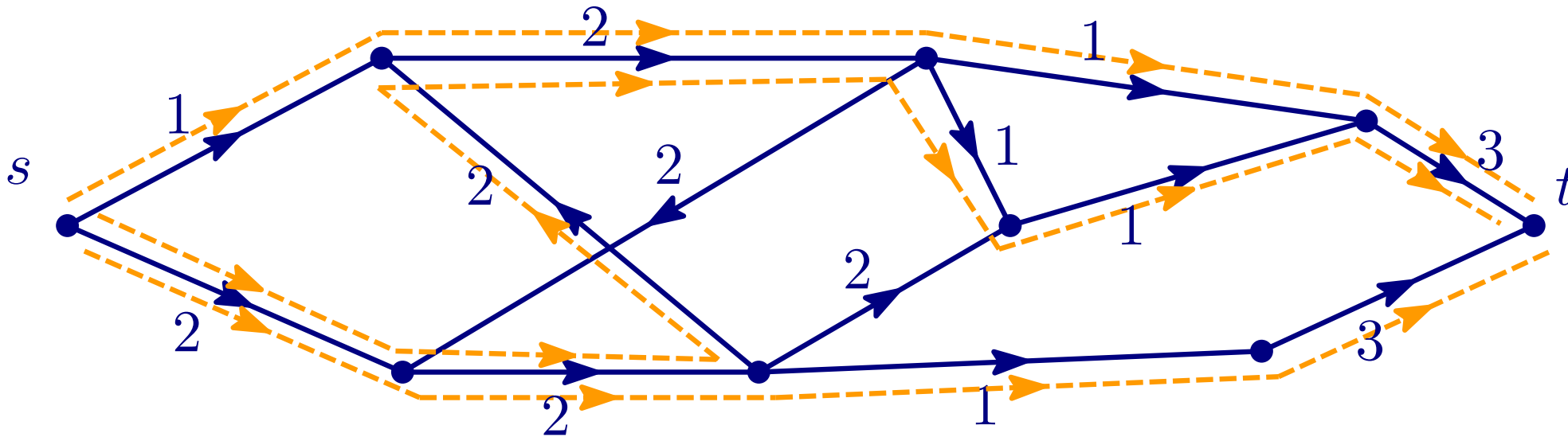
Given: Directed graph $G = (V, E)$, edge capacities $c : E \rightarrow \mathbb{Z}_{\geq 1}$, source s , and sink t .

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”

$$|f| = 3$$



Maximum Flow — A Brief History

1955	Ford-Fulkerson	$O(E \cdot \ \text{answer}\)$	\tilde{O} : hides polylog \hat{O} : hides $n^{o(1)}$
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 $\tilde{O}(EV^{2/3})$	

Maximum Flow — A Brief History

1955	Ford-Fulkerson	$O(E \cdot \ \text{answer}\)$	\tilde{O} : hides polylog \hat{O} : hides $n^{o(1)}$
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	$\hat{O}(E)$	

Maximum Flow — A Brief History

1955	Ford-Fulkerson	$O(E \cdot \ \text{answer}\)$	\tilde{O} : hides polylog \hat{O} : hides $n^{o(1)}$
1970	Edmonds-Karp	$O(VE^2)$	
1970	Dinic “Blocking Flow”	$O(V^2E)$	
1978	Malhotra	Combinatorial, Augmenting Paths “Nice”, “Simple”, “Works well in practice”	
1983	Dinic’s		
1986	Goldberg		
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	$\hat{O}(E)$	

Maximum Flow — A Brief History

1955	Ford-Fulkerson	$O(E \cdot \text{answer})$	
1970	Edmonds-Karp	$O(VE^2)$	\tilde{O} : hides polylog
1970	Dinic “Blocking Flow”	$O(V^2E)$	\hat{O} : hides $n^{o(1)}$
1978	Malhotra	Combinatorial, Augmenting Paths “Nice”, “Simple”, “Works well in practice”	
1983	Dinic’s		
1986	Goldberg		
1988	Improved “Push-Relabel”	$\tilde{O}(VE)$	
1998	Goldberg-Rao’	$\tilde{O}(E\sqrt{E})$ and $\tilde{O}(EV^{2/3})$	
2014	Lee-Sidford	$\tilde{O}(E\sqrt{E})$	
2020	Kathuria	Continuous Optimization, Interior Point Methods “I don’t understand them”	
2020	BLNPSS		
2022	Chen-Kyng-Liu-Teng-Gutierrez-Jacobs		
		$O(E)$	

-capacity)

Can we get better **combinatorial** algorithms?

New Era? — Comeback of Combinatorial Algorithms

2024

Bernstein-**B.**-Saranurak-Tu

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

“almost linear time in dense graphs”

New Era? — Comeback of Combinatorial Algorithms

2024	Bernstein- B. -Saranurak-Tu	$n^{2+o(1)}$
This Paper	Bernstein- B. -Li-Saranurak-Tu	$\tilde{O}(n^2)$ “much simpler, implementable”

Main Result: Maximum flow in on n -vertex graphs in $O(n^2 \log^{19} n)$ time.

New Era? — Comeback of Combinatorial Algorithms

2024	Bernstein- B. -Saranurak-Tu	$n^{2+o(1)}$	≈ 100 pages
This Paper	Bernstein- B. -Li-Saranurak-Tu	$\tilde{O}(n^2)$	≈ 40 pages “much simpler, implementable”

Main Result: Maximum flow in on n -vertex graphs in $O(n^2 \log^{19} n)$ time.

New Era? — Comeback of Combinatorial Algorithms

2024	Bernstein- B. -Saranurak-Tu	$n^{2+o(1)}$	≈ 100 pages
This Paper	Bernstein- B. -Li-Saranurak-Tu	$\tilde{O}(n^2)$	≈ 40 pages “much simpler, implementable”

Main Result: Maximum flow in on n -vertex graphs in $O(n^2 \log^{19} n)$ time.

After $\log(n)$ simplifications we will have a SOSA paper :)

New Era? — Comeback of Combinatorial Algorithms

2024	Bernstein- B. -Saranurak-Tu	$n^{2+o(1)}$	≈ 100 pages
This Paper	Bernstein- B. -Li-Saranurak-Tu	$\tilde{O}(n^2)$	≈ 40 pages “much simpler, implementable”

Main Result: Maximum flow in on n -vertex graphs in $O(n^2 \log^{19} n)$ time.

<https://github.com/TaWeiTu/combinatorial-max-flow>

Full implementation!

New Era? — Comeback of Combinatorial Algorithms

2024	Bernstein- B. -Saranurak-Tu	$n^{2+o(1)}$	≈ 100 pages
This Paper	Bernstein- B. -Li-Saranurak-Tu	$\tilde{O}(n^2)$	≈ 40 pages “much simpler, implementable”

Main Result: Maximum flow in on n -vertex graphs in $O(n^2 \log^{19} n)$ time.

<https://github.com/TaWeiTu/combinatorial-max-flow>

Full implementation!

Runs in $\ll 1$ sec for $n = 2$ on my laptop



New Era? — Comeback of Combinatorial Algorithms

2024	Bernstein- B. -Saranurak-Tu	$n^{2+o(1)}$	≈ 100 pages
This Paper	Bernstein- B. -Li-Saranurak-Tu	$\tilde{O}(n^2)$	≈ 40 pages “much simpler, implementable”

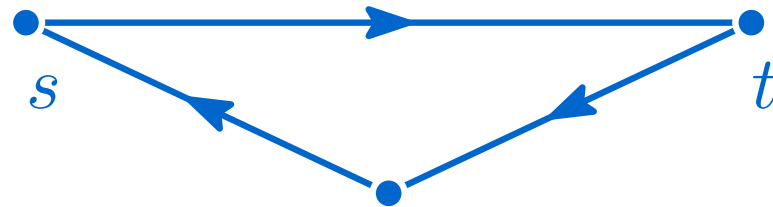
Main Result: Maximum flow in on n -vertex graphs in $O(n^2 \log^{19} n)$ time.

<https://github.com/TaWeiTu/combinatorial-max-flow>

Full implementation!

Runs in $\ll 1$ sec for $n = 2$ on my laptop

$n = 3$ is still running on my laptop...



New Era? — Comeback of Combinatorial Algorithms

2024	Bernstein- B. -Saranurak-Tu	$n^{2+o(1)}$	≈ 100 pages
This Paper	Bernstein- B. -Li-Saranurak-Tu	$\tilde{O}(n^2)$	≈ 40 pages “much simpler, implementable”

Main Result: Maximum flow in on n -vertex graphs in $O(n^2 \log^{19} n)$ time.

<https://github.com/TaWeiTu/combinatorial-max-flow>

Full implementation!

(Setting parameters slightly inaccurately: $n \approx 50$, $m \approx 1000$ in 1 sec.)

Theoretically possible \mapsto Implementable \mapsto Practical

New Era? — Comeback of Combinatorial Algorithms

2024	Bernstein- B. -Saranurak-Tu	$n^{2+o(1)}$	≈ 100 pages
This Paper	Bernstein- B. -Li-Saranurak-Tu	$\tilde{O}(n^2)$	≈ 40 pages “much simpler, implementable”

Main Result: Maximum flow in on n -vertex graphs in $O(n^2 \log^{19} n)$ time.

Techniques:

Augmenting Paths (new version of Push-Relabel)

Directed Expander Hierarchy

New: Shortcuts

New Era? — Comeback of Combinatorial Algorithms

2024	Bernstein- B. -Saranurak-Tu	$n^{2+o(1)}$	≈ 100 pages
This Paper	Bernstein- B. -Li-Saranurak-Tu	$\tilde{O}(n^2)$	≈ 40 pages “much simpler, implementable”

Main Result: Maximum flow in on n -vertex graphs in $O(n^2 \log^{19} n)$ time.

Techniques:

Augmenting Paths (new version of Push-Relabel)

Directed Expander Hierarchy

New: Shortcuts

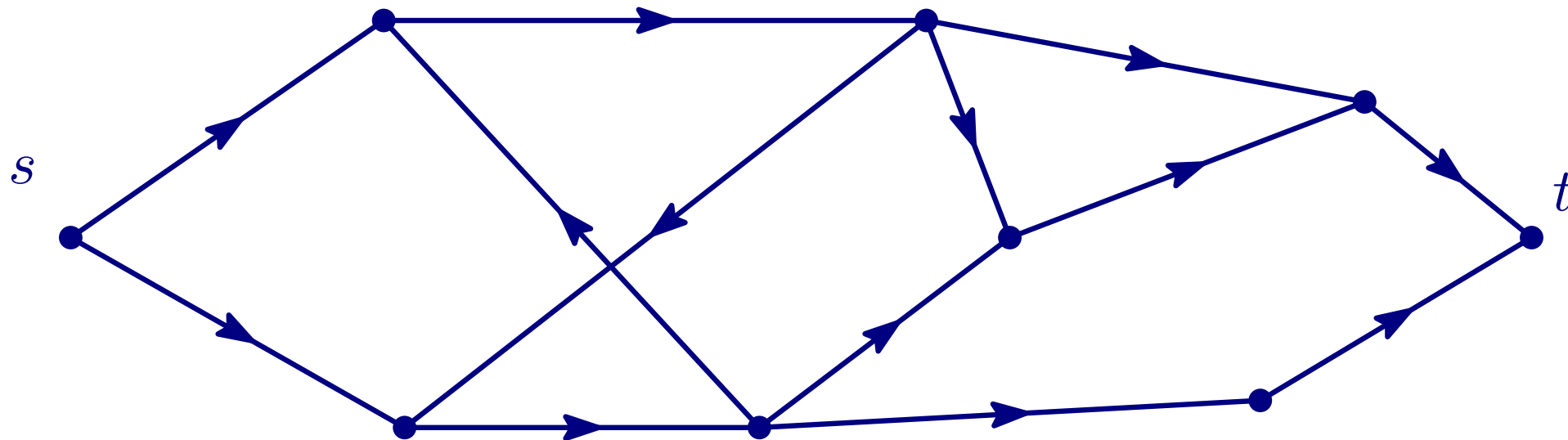
My Hope: (in a few years)

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

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

Augmenting Paths

[Ford-Fulkerson 1955]
[Jacobi 1836]

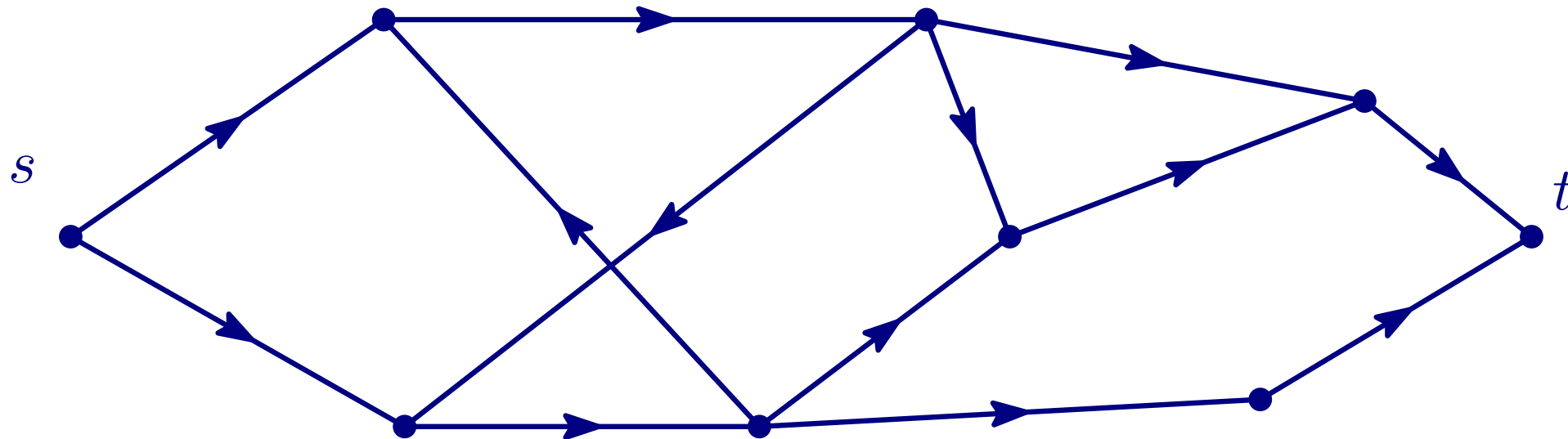


Augmenting Paths

[Ford-Fulkerson 1955]

[Jacobi 1836]

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



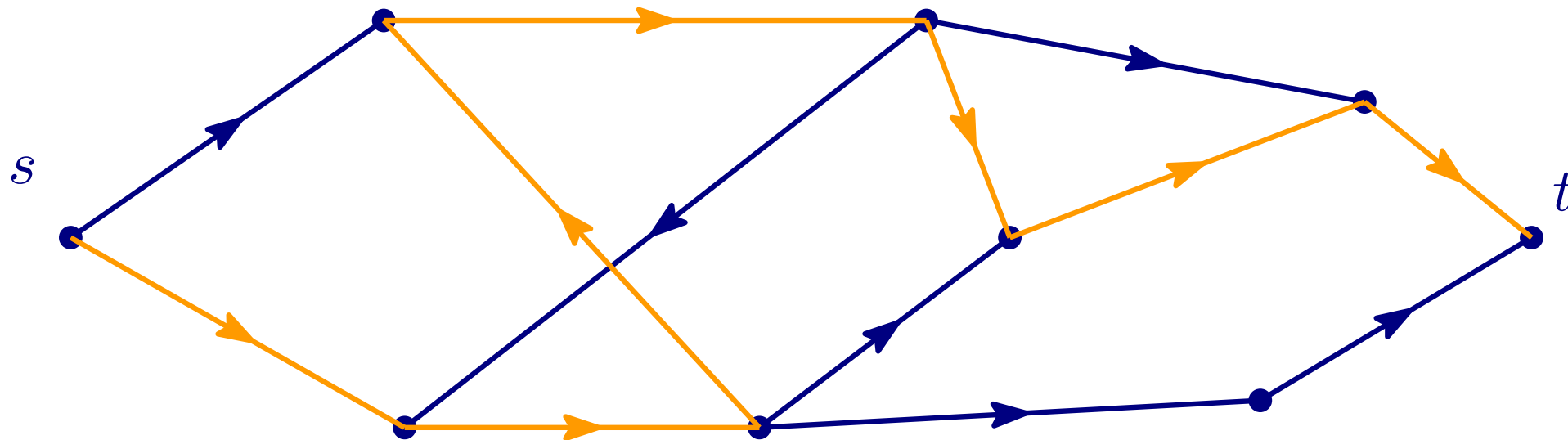
Augmenting Paths

[Ford-Fulkerson 1955]

[Jacobi 1836]

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

Augmenting Path

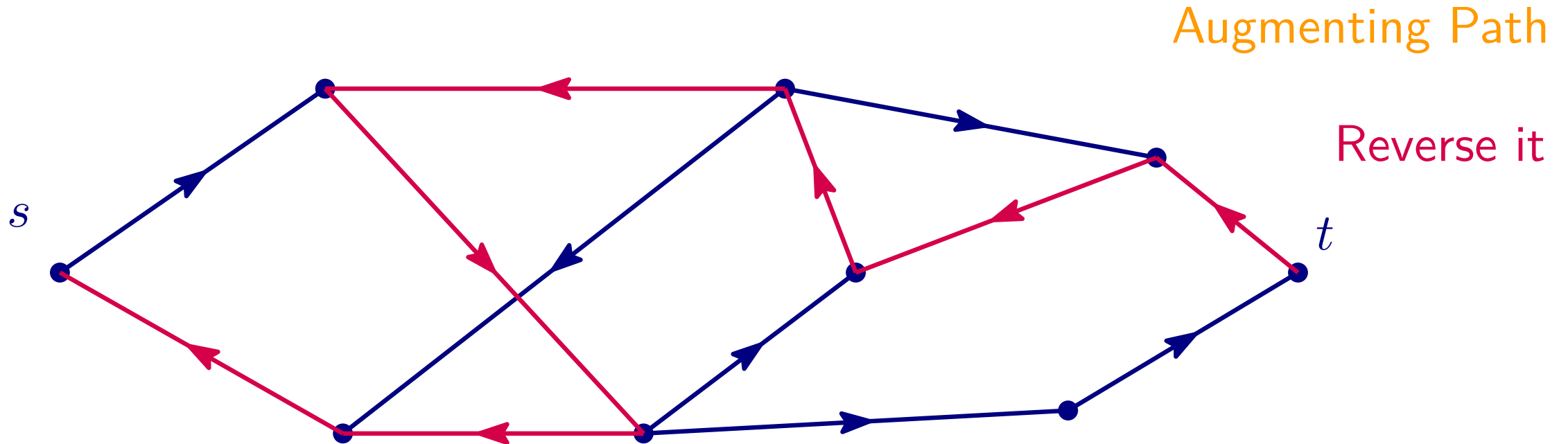


Augmenting Paths

[Ford-Fulkerson 1955]

[Jacobi 1836]

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

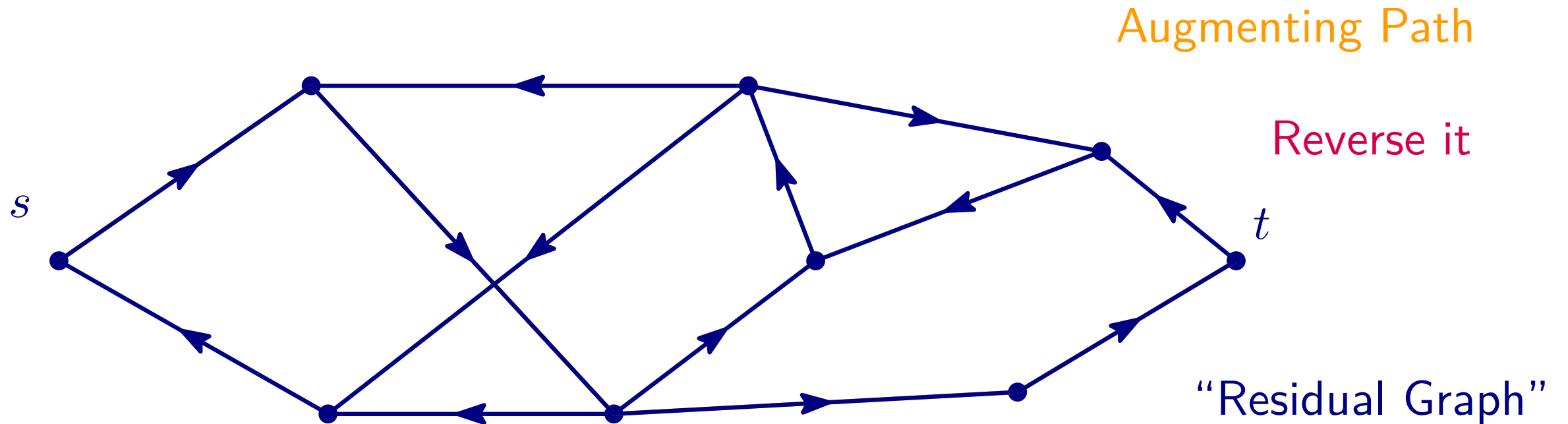


Augmenting Paths

[Ford-Fulkerson 1955]

[Jacobi 1836]

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

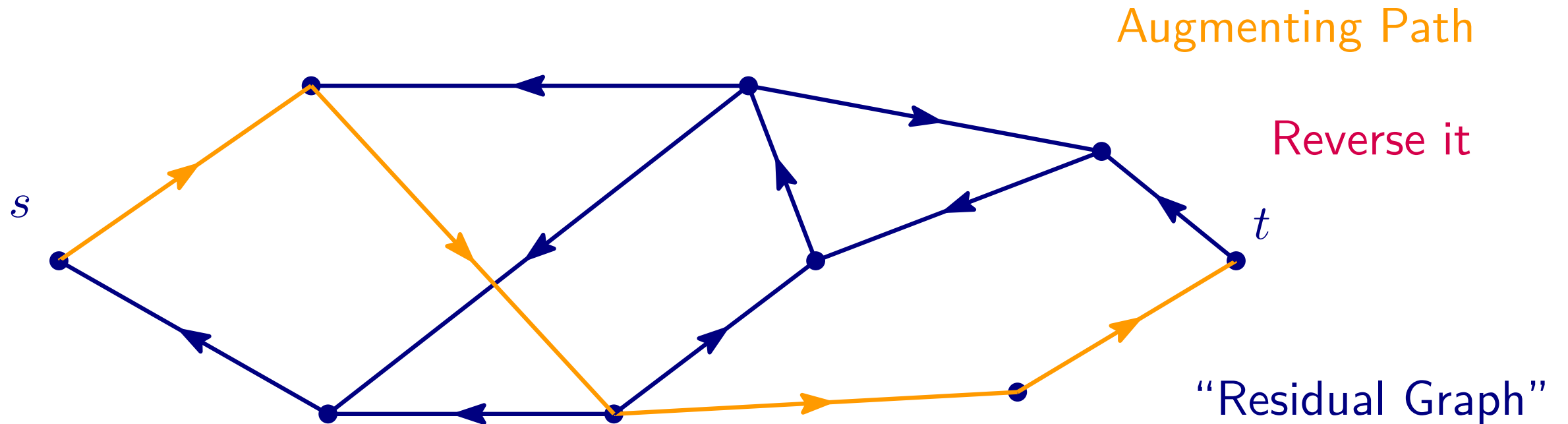


Augmenting Paths

[Ford-Fulkerson 1955]

[Jacobi 1836]

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

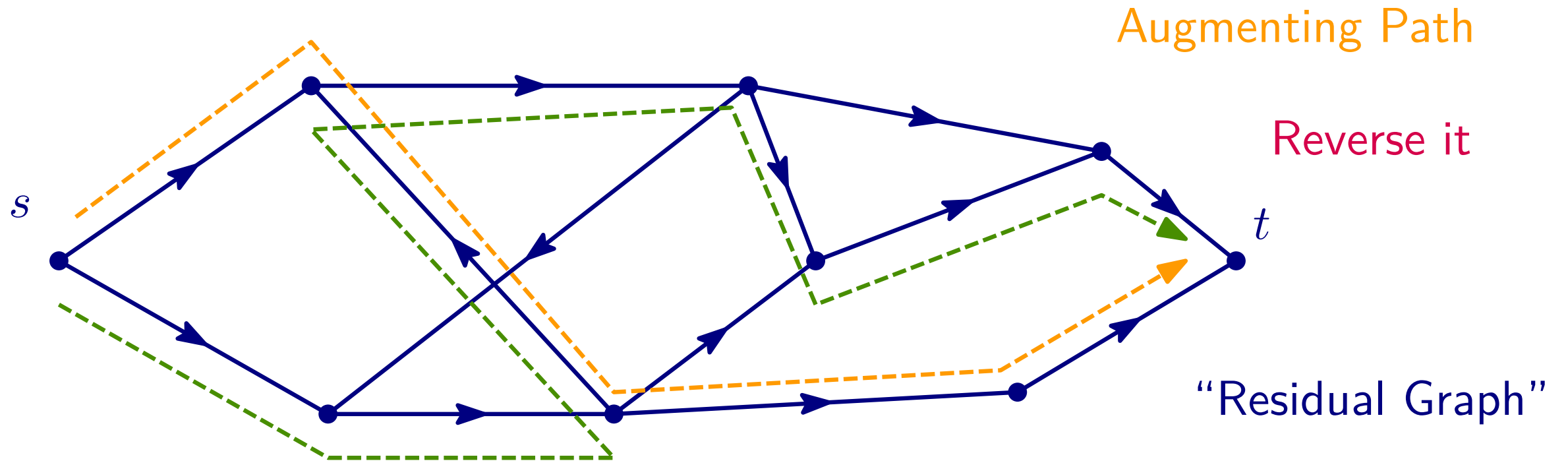


Augmenting Paths

[Ford-Fulkerson 1955]

[Jacobi 1836]

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

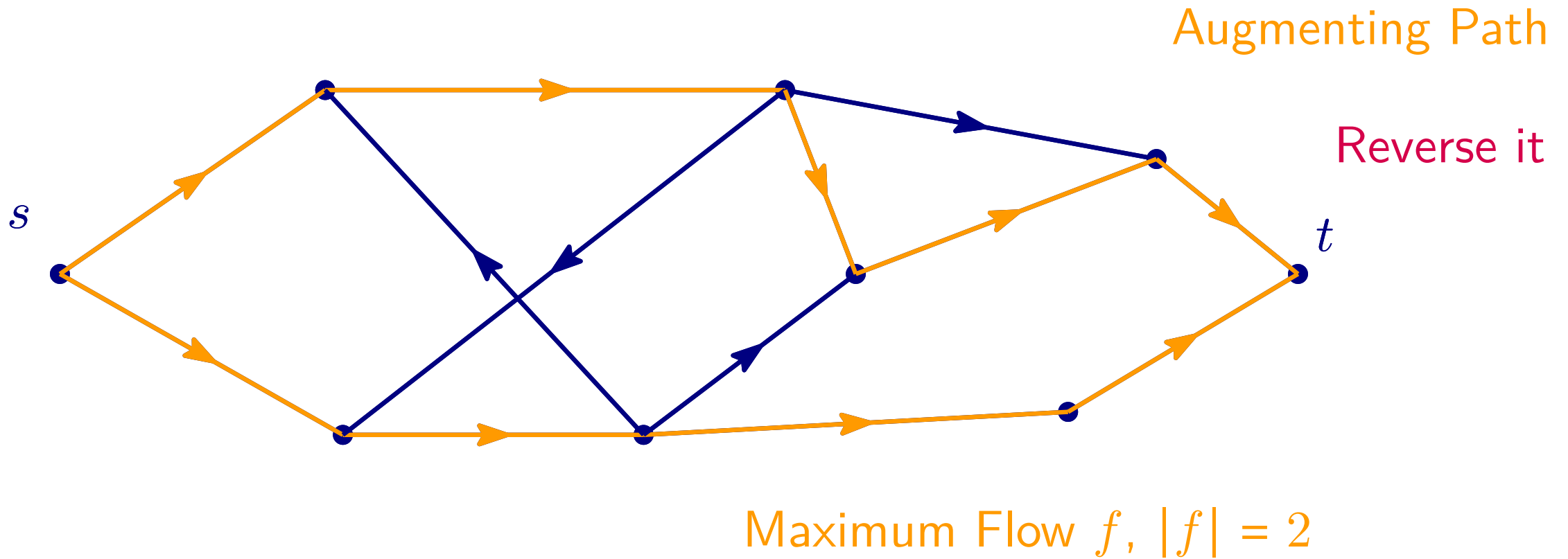


Augmenting Paths

[Ford-Fulkerson 1955]

[Jacobi 1836]

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



Approximate Flow \implies Exact Flow

Approximate Flow \implies Exact Flow

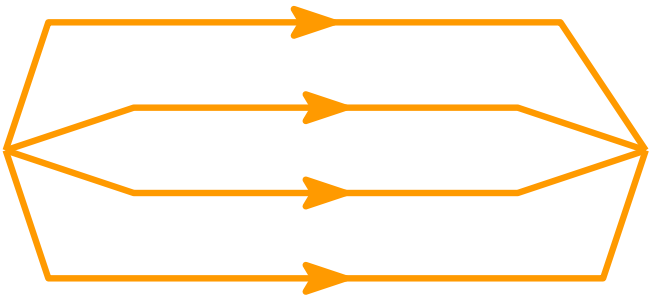
Proof. Recurse on residual graph.

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

(does not work in undirected graphs)

Short Flow

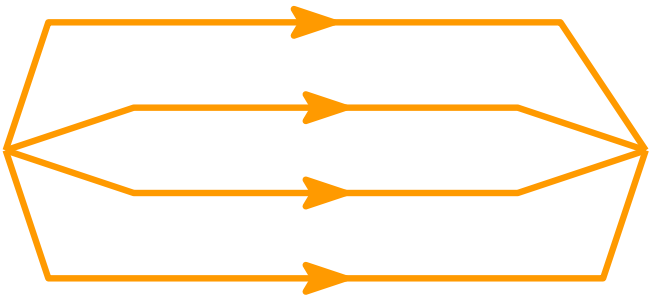
Guarantee: Suppose max-flow f^* of G is short



Short Flow

Guarantee: Suppose max-flow f^* of G is short

average length of flow-path is $\leq h$ ($\approx \text{polylog}(n)$)

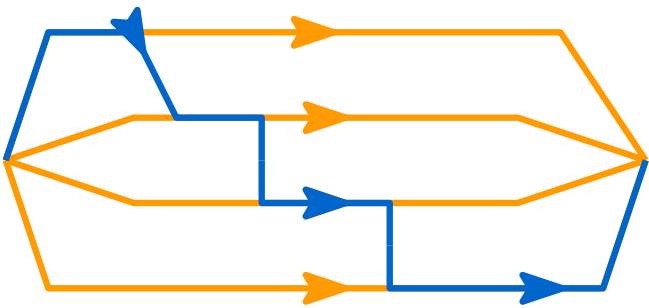


Short Flow

Guarantee: Suppose max-flow f^* of G is short

average length of flow-path is $\leq h$ ($\approx \text{polylog}(n)$)

Theorem: Can compute 10-approx max-flow f with $|f| \geq 0.1 \cdot |f^*|$ in $\tilde{O}(mh)$ time



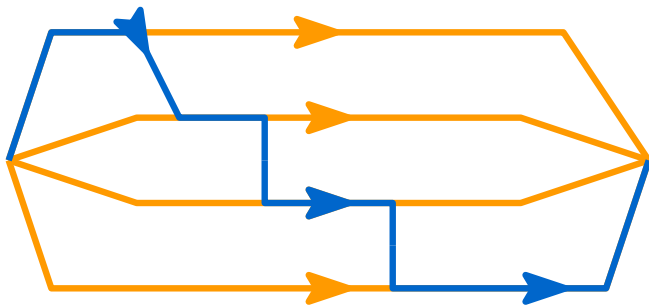
Short Flow

Guarantee: Suppose max-flow f^* of G is short

average length of flow-path is $\leq h$ ($\approx \text{polylog}(n)$)

Theorem: Can compute 10-approx max-flow f with $|f| \geq 0.1 \cdot |f^*|$ in $\tilde{O}(mh)$ time

Proof: Run Push-Relabel / Dinic's Blocking-Flow up to depth $2h$.



Short Flow

Guarantee: Suppose max-flow f^* of G is short

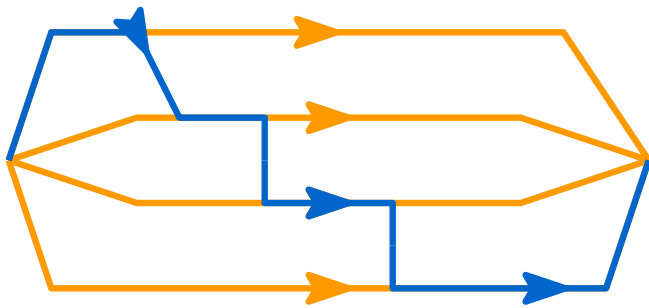
average length of flow-path is $\leq h$ ($\approx \text{polylog}(n)$)

Theorem: Can compute 10-approx max-flow f with $|f| \geq 0.1 \cdot |f^*|$ in $\tilde{O}(mh)$ time

Proof: Run Push-Relabel / Dinic's Blocking-Flow up to depth $2h$.

(1): lengths of flow path in f is $\leq 2h$

(2): shortest s - t path in residual graph G_f length $> 2h$



Short Flow

Guarantee: Suppose max-flow f^* of G is short

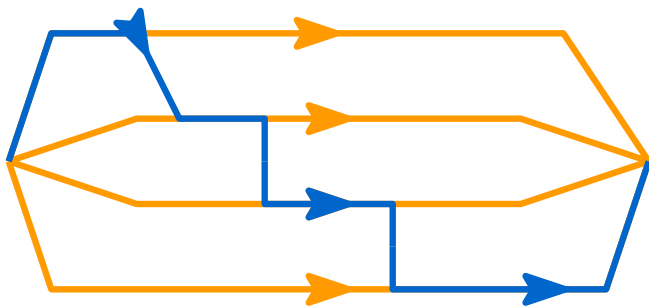
average length of flow-path is $\leq h$ ($\approx \text{polylog}(n)$)

Theorem: Can compute 10-approx max-flow f with $|f| \geq 0.1 \cdot |f^*|$ in $\tilde{O}(mh)$ time

Proof: Run Push-Relabel / Dinic's Blocking-Flow up to depth $2h$.

(1): lengths of flow path in f is $\leq 2h$

(2): shortest s - t path in residual graph G_f length $> 2h$



Short Flow

Guarantee: Suppose max-flow f^* of G is short

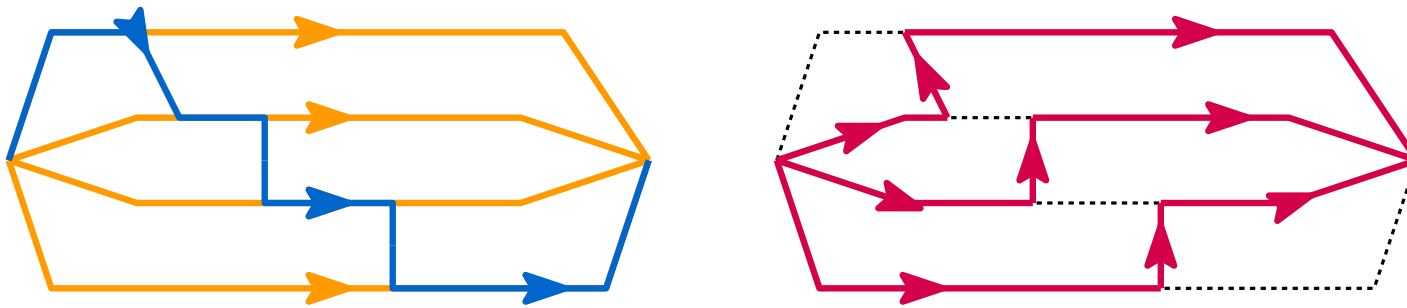
average length of flow-path is $\leq h$ ($\approx \text{polylog}(n)$)

Theorem: Can compute 10-approx max-flow f with $|f| \geq 0.1 \cdot |f^*|$ in $\tilde{O}(mh)$ time

Proof: Run Push-Relabel / Dinic's Blocking-Flow up to depth $2h$.

(1): lengths of flow path in f is $\leq 2h$

(2): shortest s - t path in residual graph G_f length $> 2h$



$f^* - f$ is flow in residual graph

Short Flow

Guarantee: Suppose max-flow f^* of G is short

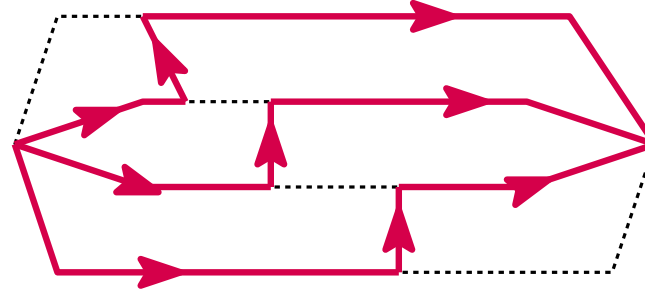
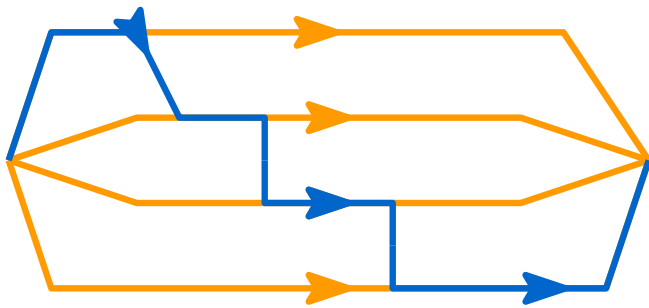
average length of flow-path is $\leq h$ ($\approx \text{polylog}(n)$)

Theorem: Can compute 10-approx max-flow f with $|f| \geq 0.1 \cdot |f^*|$ in $\tilde{O}(mh)$ time

Proof: Run Push-Relabel / Dinic's Blocking-Flow up to depth $2h$.

(1): lengths of flow path in f is $\leq 2h$

(2): shortest s - t path in residual graph G_f length $> 2h$



$f^* - f$ is flow in residual graph

$$\text{average-length}(f^* - f) \leq \frac{|f^*| \cdot h + |f| \cdot 2h}{|f^*| - |f|}$$

Short Flow

Guarantee: Suppose max-flow f^* of G is short

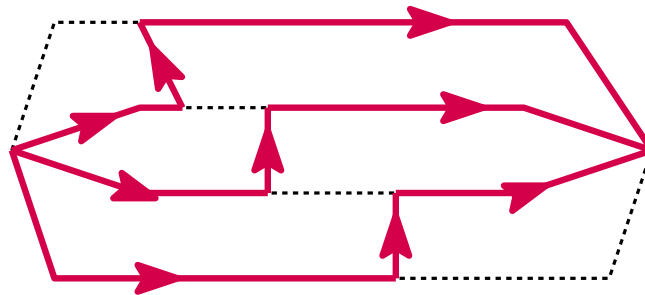
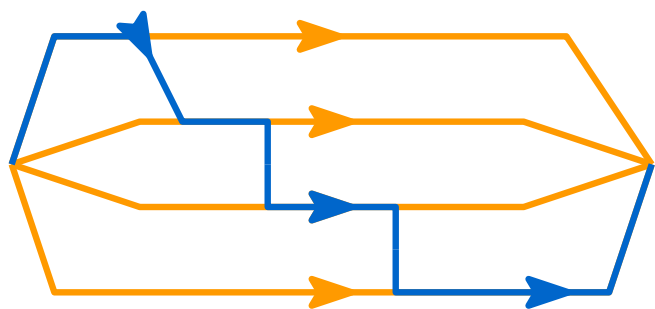
average length of flow-path is $\leq h$ ($\approx \text{polylog}(n)$)

Theorem: Can compute 10-approx max-flow f with $|f| \geq 0.1 \cdot |f^*|$ in $\tilde{O}(mh)$ time

Proof: Run Push-Relabel / Dinic's Blocking-Flow up to depth $2h$.

(1): lengths of flow path in f is $\leq 2h$

(2): shortest s - t path in residual graph G_f length $> 2h$



$f^* - f$ is flow in residual graph

$$\text{average-length}(f^* - f) \leq \frac{|f^*| \cdot h + |f| \cdot 2h}{|f^*| - |f|}$$

$$\leq \frac{1.2h \cdot |f^*|}{0.9 \cdot |f^*|} \ll 2h$$

(assuming $|f| < 0.1|f^*|$)

Short Flow

Guarantee: Suppose max-flow f^* of G is short

average length of flow-path is $\leq h$ ($\approx \text{polylog}(n)$)

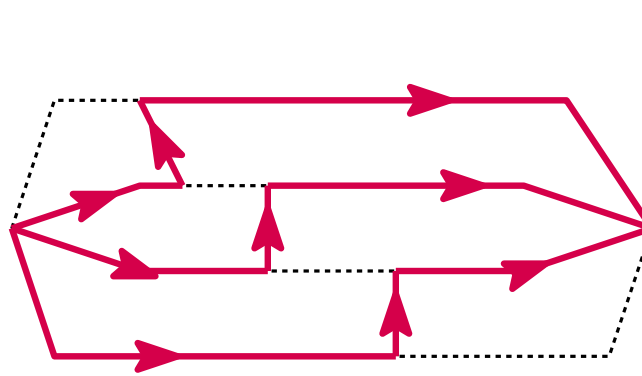
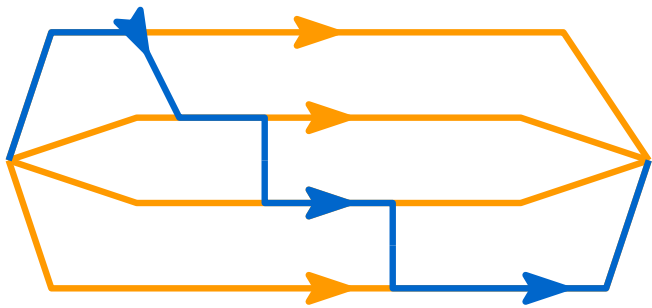
Theorem: Can compute 10-approx max-flow f with $|f| \geq 0.1 \cdot |f^*|$ in $\tilde{O}(mh)$ time

Proof: Run Push-Relabel / Dinic's Blocking-Flow up to depth $2h$.

(1): lengths of flow path in f is $\leq 2h$

(2): shortest s - t path in residual graph G_f length $> 2h$

Contradiction!



$f^* - f$ is flow in residual graph

$$\text{average-length}(f^* - f) \leq \frac{|f^*| \cdot h + |f| \cdot 2h}{|f^*| - |f|}$$

$$\leq \frac{1.2h \cdot |f^*|}{0.9 \cdot |f^*|} \ll 2h$$

(assuming $|f| < 0.1|f^*|$)

Guarantee: Suppose max-flow f^* of G is short

Theorem: Can compute 10-approx max-flow f with $|f| \geq 0.1 \cdot |f^*|$ in $\tilde{O}(mh)$ time

Guarantee: Suppose max-flow f^* of G is short for edge lengths $w(e)$

Theorem: Can compute 10-approx max-flow f with $|f| \geq 0.1 \cdot |f^*|$ in ~~$\tilde{O}(mh)$~~ time
“Weighted Push-Relabel” $O(h \cdot \sum_e \frac{1}{w(e)})$

Guarantee: Suppose max-flow f^* of G is short for edge lengths $w(e)$

Theorem: Can compute 10-approx max-flow f with $|f| \geq 0.1 \cdot |f^*|$ in $\tilde{O}(mh)$ time
“Weighted Push-Relabel”
 $O(h \cdot \sum_e \frac{1}{w(e)})$

How to find good lengths $w(e)$ that satisfies guarantee and has small $\sum_e \frac{1}{w(e)}$?

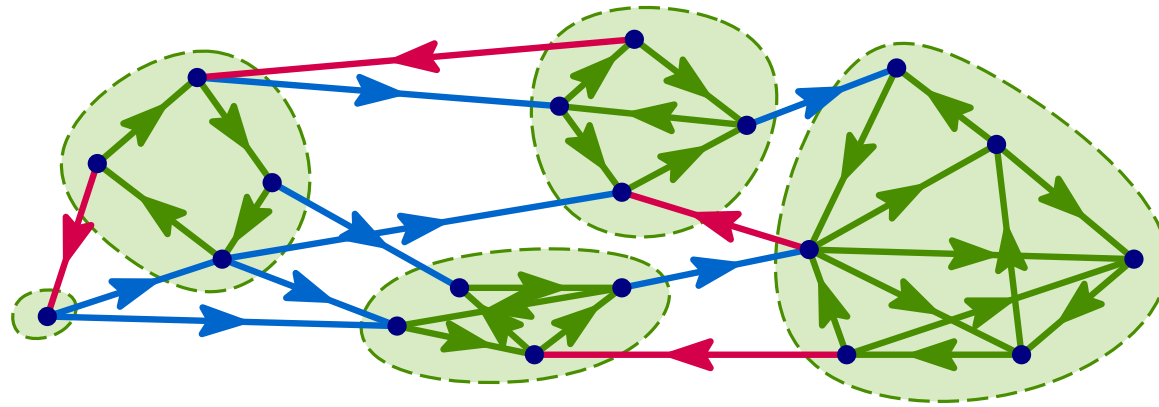
Guarantee: Suppose max-flow f^* of G is **short** **for edge lengths** $w(e)$

Theorem: Can compute 10-approx max-flow f with $|f| \geq 0.1 \cdot |f^*|$ in ~~$\tilde{O}(mh)$~~ time
“Weighted Push-Relabel”
 $O(h \cdot \sum_e \frac{1}{w(e)})$

How to find *good* lengths $w(e)$ that satisfies guarantee and has small $\sum_e \frac{1}{w(e)}$?

Directed Expander Hierarchy

Graph = DAG + Expanders + Few back-edges



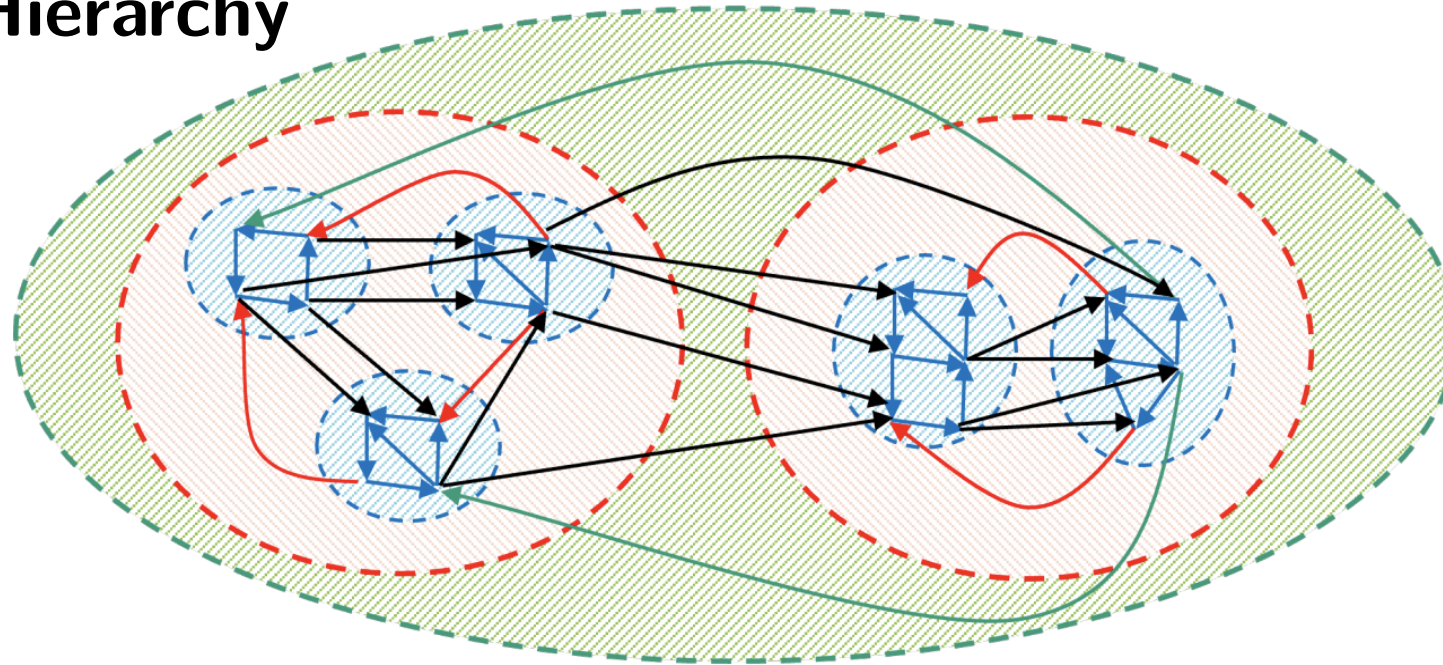
Guarantee: Suppose max-flow f^* of G is **short** **for edge lengths** $w(e)$

Theorem: Can compute 10-approx max-flow f with $|f| \geq 0.1 \cdot |f^*|$ in ~~$\tilde{O}(mh)$~~ time
“Weighted Push-Relabel” $O(h \cdot \sum_e \frac{1}{w(e)})$

How to find *good* lengths $w(e)$ that satisfies guarantee and has small $\sum_e \frac{1}{w(e)}$?

“topological” $w(u, v) = |\tau(u) - \tau(v)|$

Directed Expander Hierarchy



Guarantee: Suppose max-flow f^* of G is short for edge lengths $w(e)$

Theorem: Can compute 10-approx max-flow f with $|f| \geq 0.1 \cdot |f^*|$ in $\tilde{O}(mh)$ time
“Weighted Push-Relabel”
 $O(h \cdot \sum_e \frac{1}{w(e)})$

How to find *good* lengths $w(e)$ that satisfies guarantee and has small $\sum_e \frac{1}{w(e)}$?

“topological” $w(u, v) = |\tau(u) - \tau(v)|$

Directed Expander Hierarchy

Really annoying
to construct
 ≈ 50 pages

7	Building an Expander Hierarchy	49
7.1	Overview and Setup	49
7.1.1	Intuition of Analysis	50
7.1.2	A Data Structure Point-of-View	51
7.1.3	Overview of [HKPW23]	53
7.2	Constructing and Repairing Witnesses	55
7.3	Stability of Witnesses	59
7.4	Maintaining Expander Decomposition	64
7.5	Bounding Expected Recourse and Running Time	71
7.5.1	Solving the Recurrences	78
7.6	Putting Everything Together	88

New Idea: Shortcuts

Guarantee: Suppose max-flow f^* of G is short

How to guarantee this?

New Idea: Shortcuts

Guarantee: Suppose max-flow f^* of G is short

How to guarantee this?

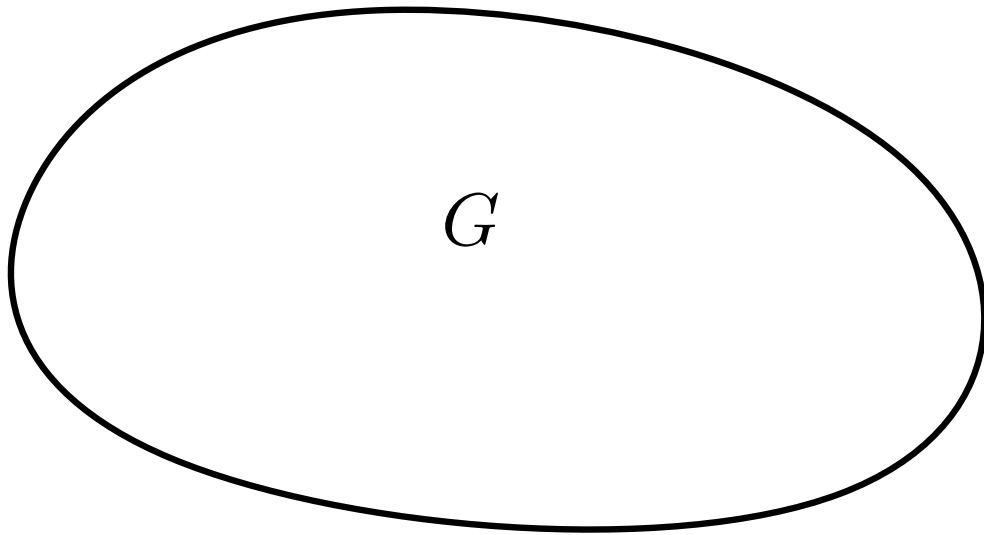
Answer: We will cheat!

New Idea: Shortcuts

Guarantee: Suppose max-flow f^* of G is short

How to guarantee this?

Answer: We will cheat!

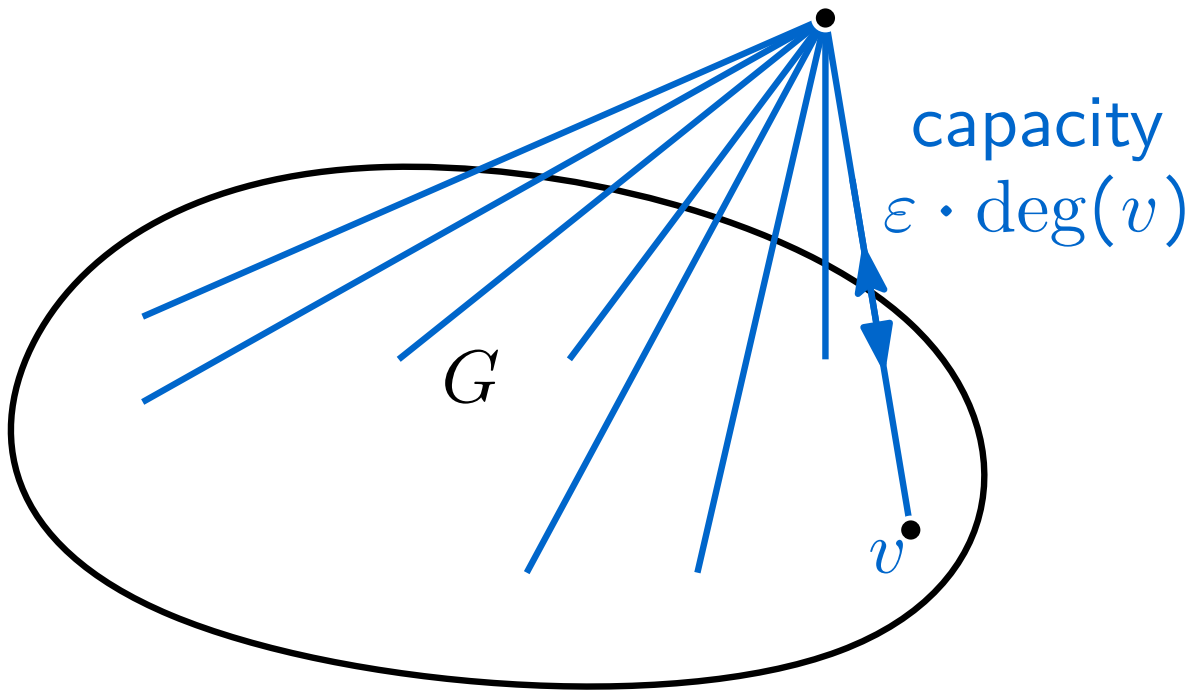


New Idea: Shortcuts

Guarantee: Suppose max-flow f^* of G is short

How to guarantee this?

Answer: We will cheat! “Shortcut Star”

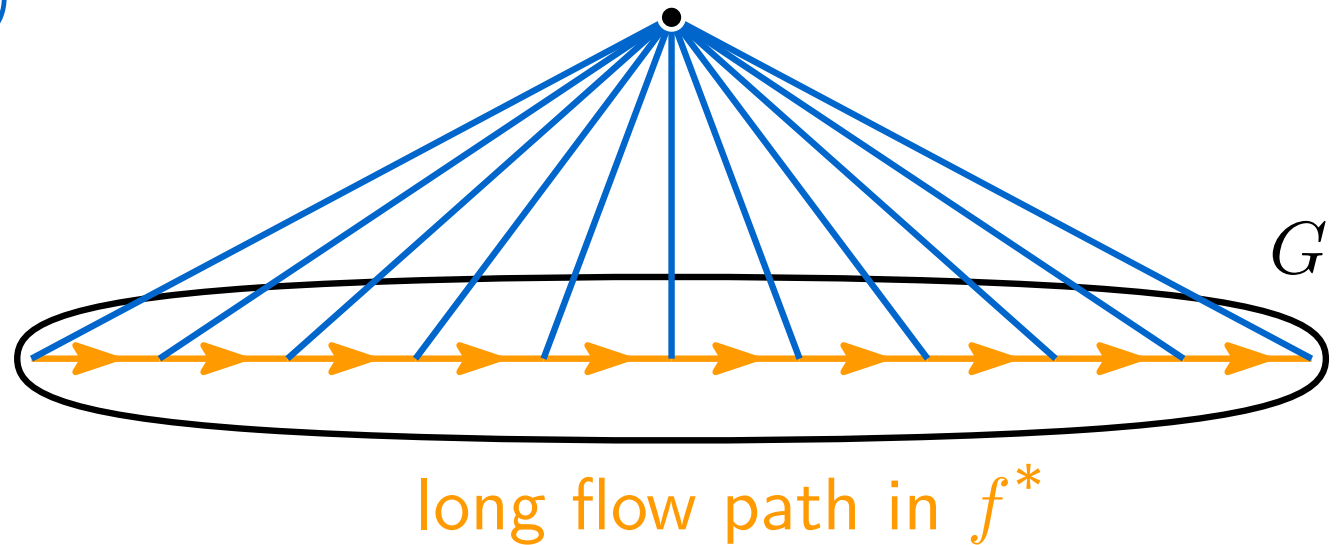
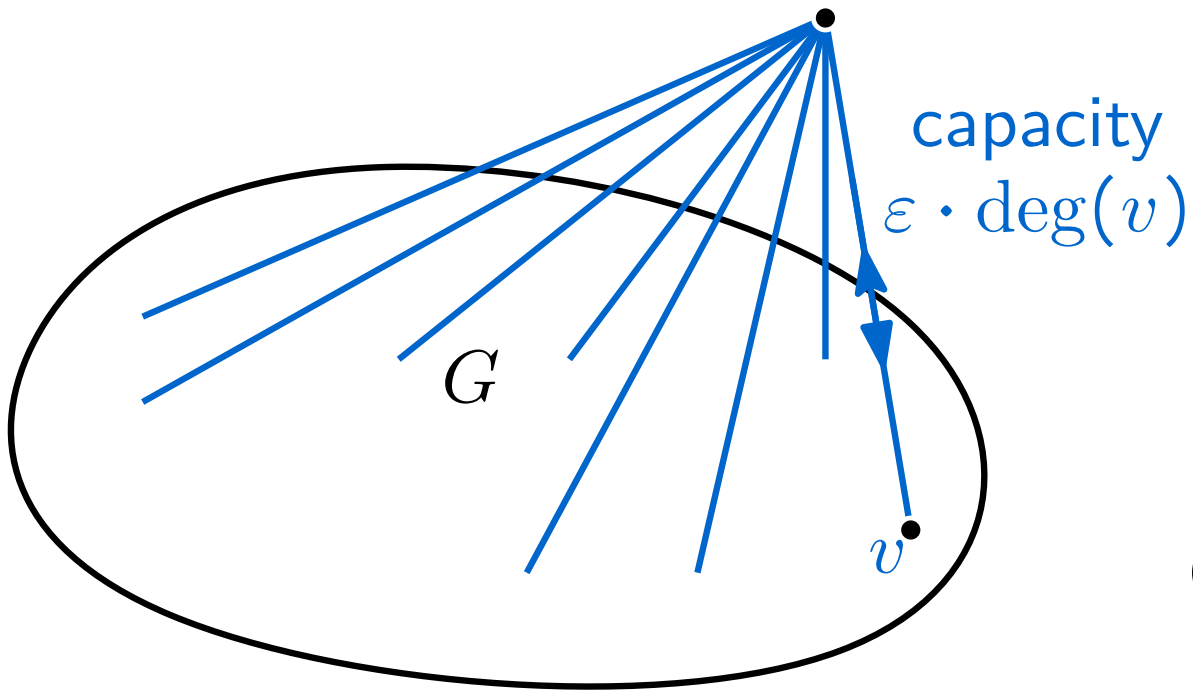


New Idea: Shortcuts

Guarantee: Suppose max-flow f^* of G is short

How to guarantee this?

Answer: We will cheat! “Shortcut Star”

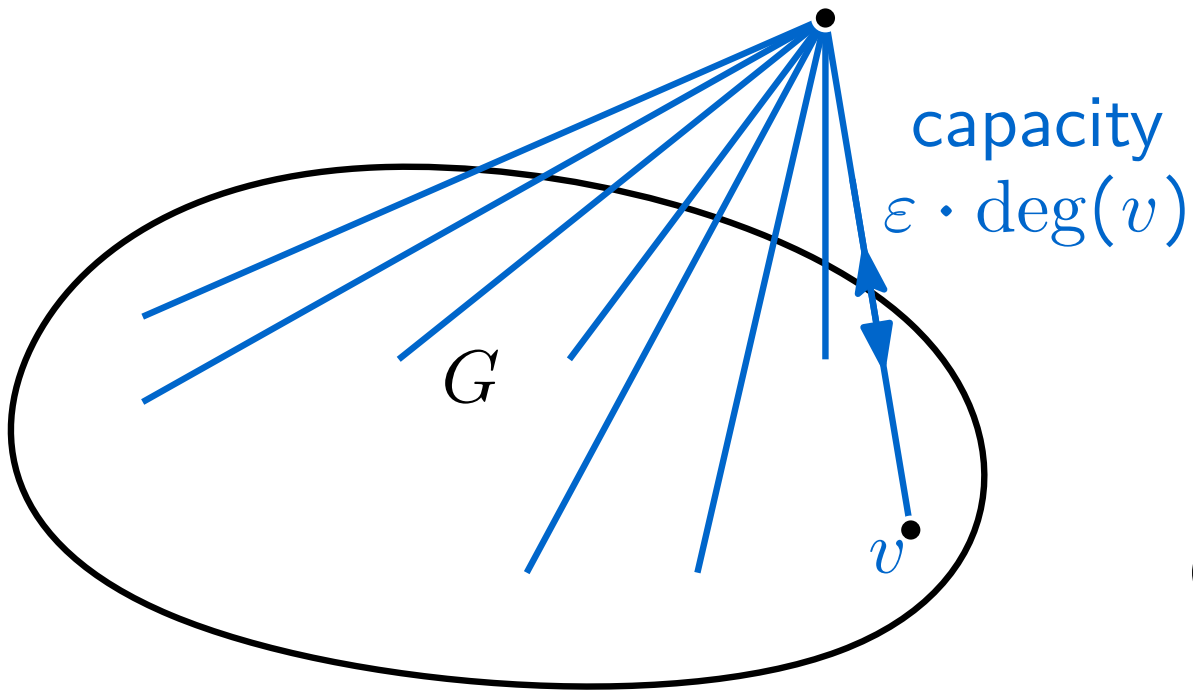


New Idea: Shortcuts

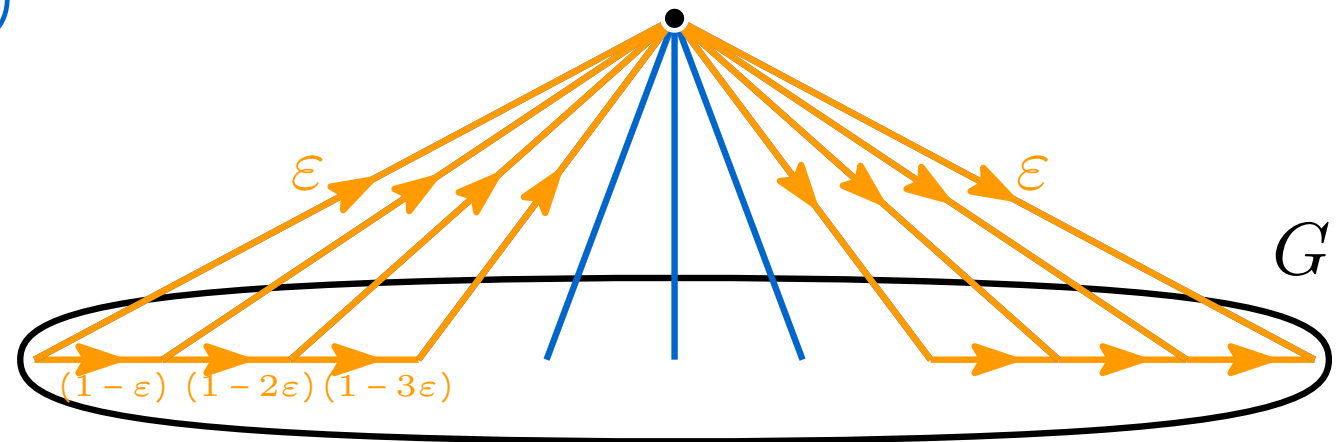
Guarantee: Suppose max-flow f^* of G is short

How to guarantee this?

Answer: We will cheat! “Shortcut Star”



“Leak” flow in first / last $\frac{1}{\varepsilon}$ vertices



long flow path in f^*

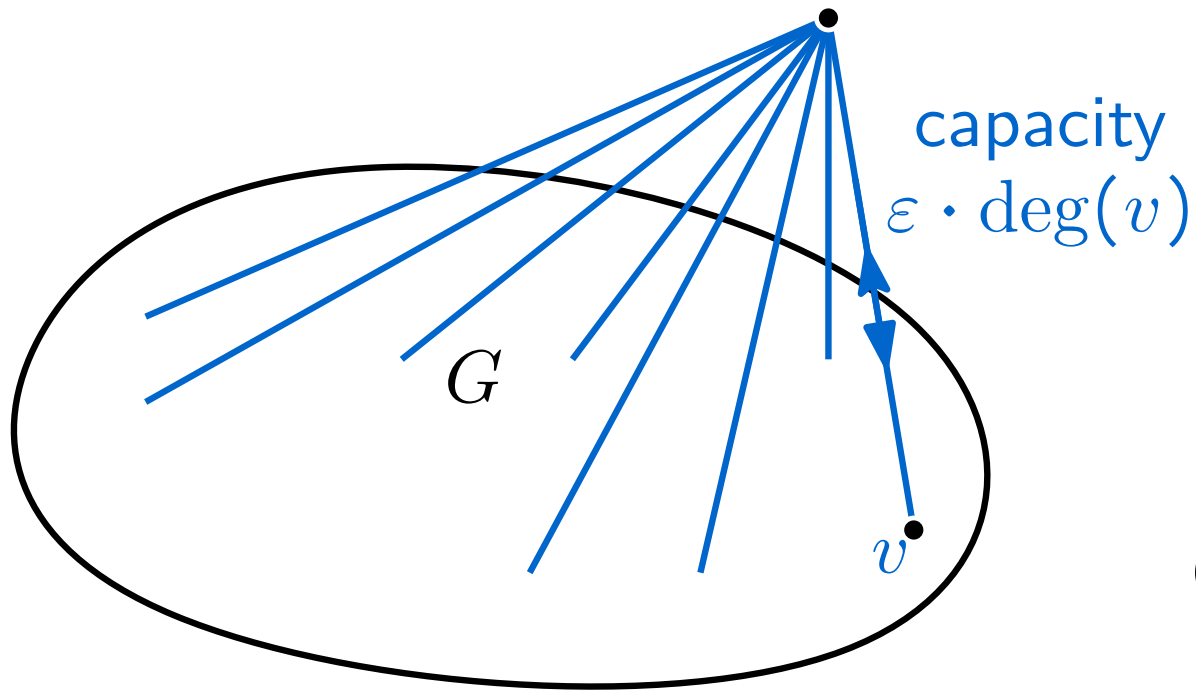
New Idea: Shortcuts

Guarantee: Suppose max-flow f^* of G is short

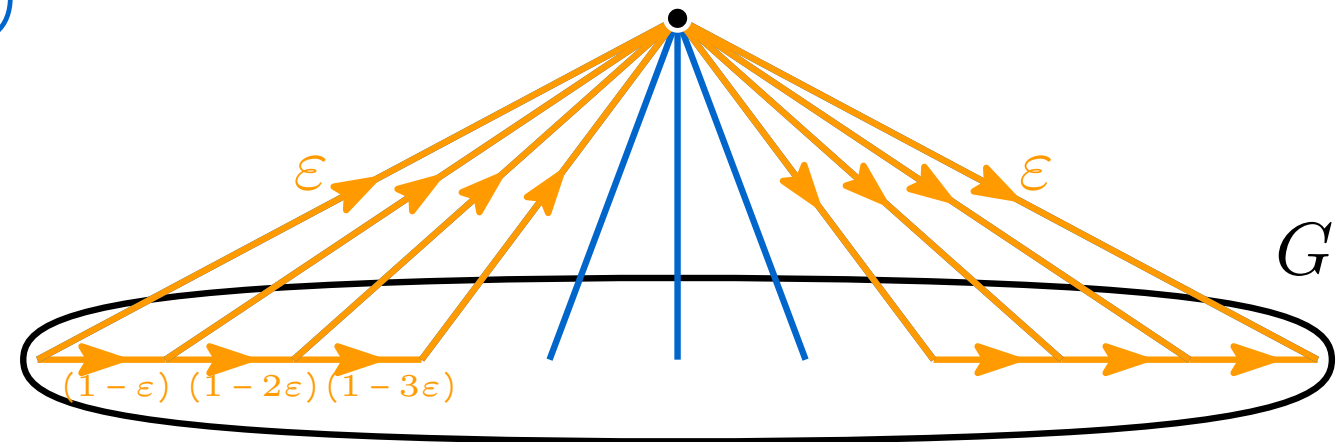
Guarantee satisfied in $G + \text{star}$

How to guarantee this?

Answer: We will cheat! “Shortcut Star”



“Leak” flow in first / last $\frac{1}{\varepsilon}$ vertices



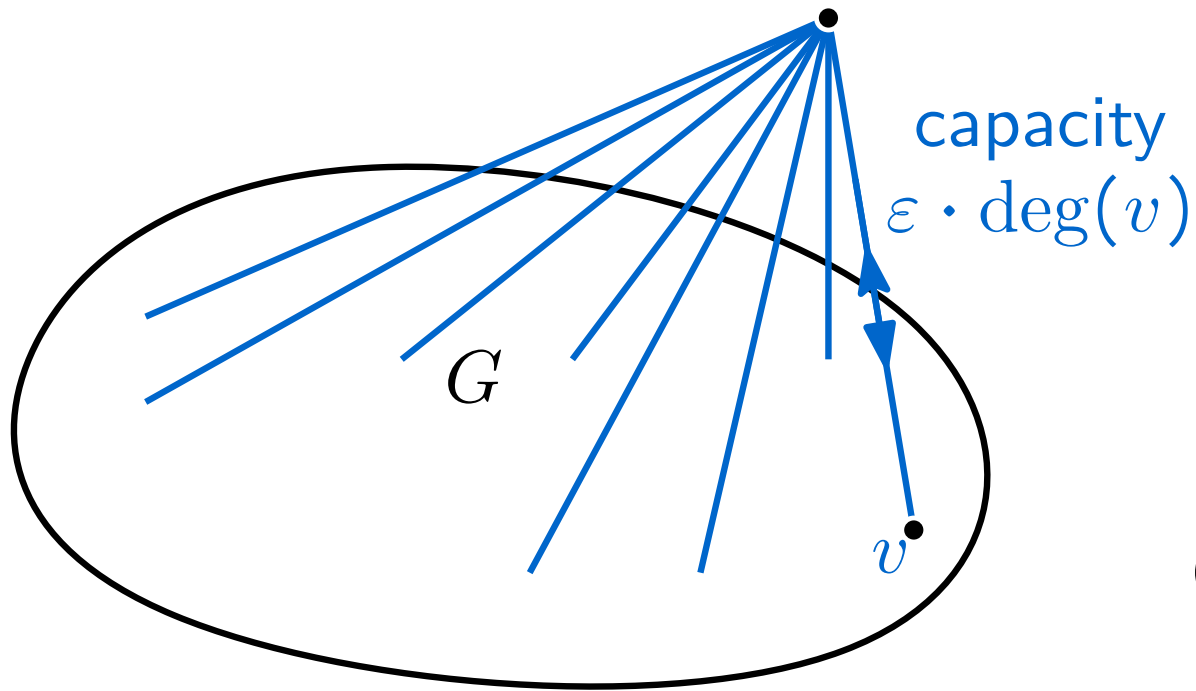
long flow path in f^*

New Idea: Shortcuts

Guarantee: Suppose max-flow f^* of G is short

How to guarantee this?

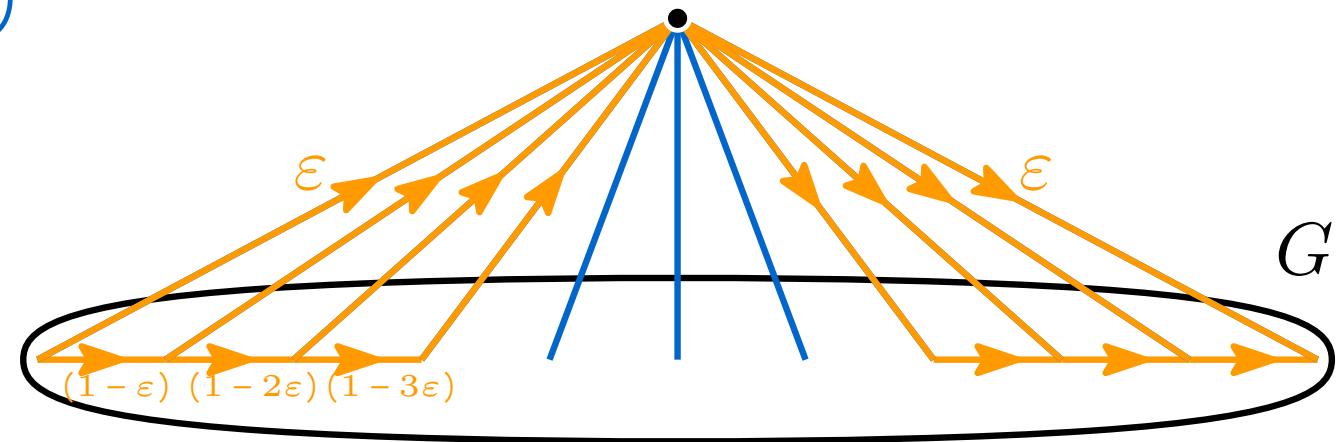
Answer: We will cheat! “Shortcut Star”



Guarantee satisfied in $G + \text{star}$

Compute max flow in $G + \text{star}$ in $\tilde{O}(m/\varepsilon)$ time!

“Leak” flow in first / last $\frac{1}{\varepsilon}$ vertices



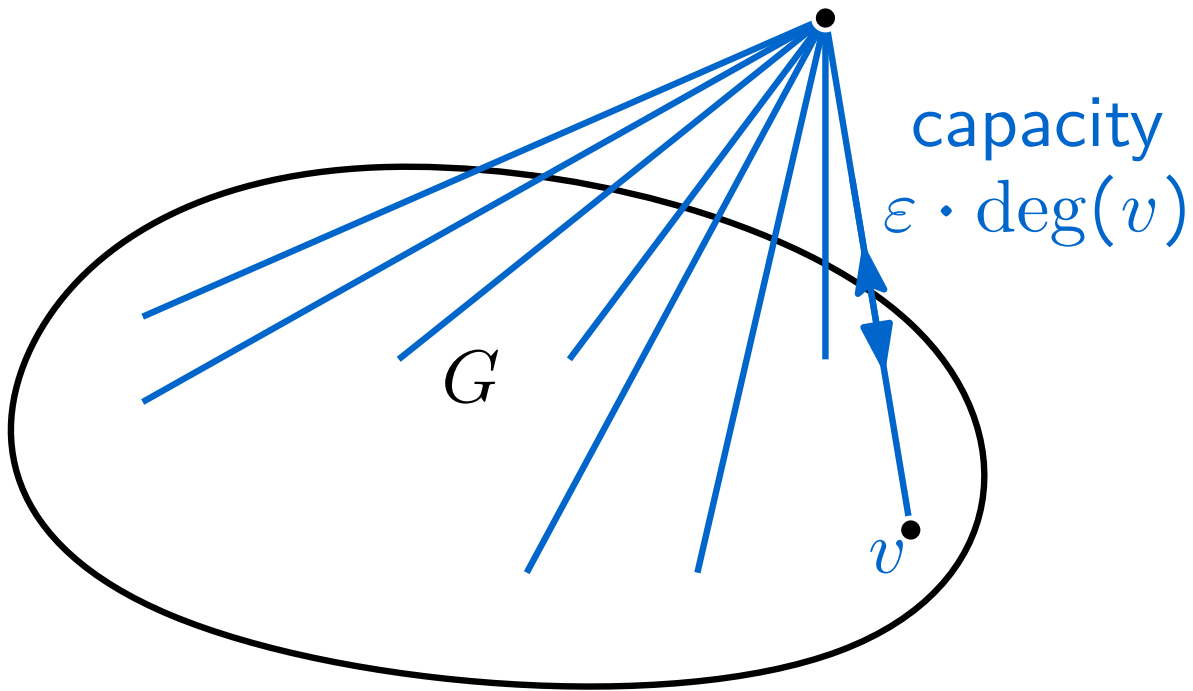
long flow path in f^*

New Idea: Shortcuts

Guarantee: Suppose max-flow f^* of G is **short**

How to guarantee this?

Answer: We will cheat! “Shortcut Star”



Guarantee satisfied in $G + \text{star}$

Compute max flow in $G + \text{star}$ in $\tilde{O}(m/\epsilon)$ time!

Big Problem:

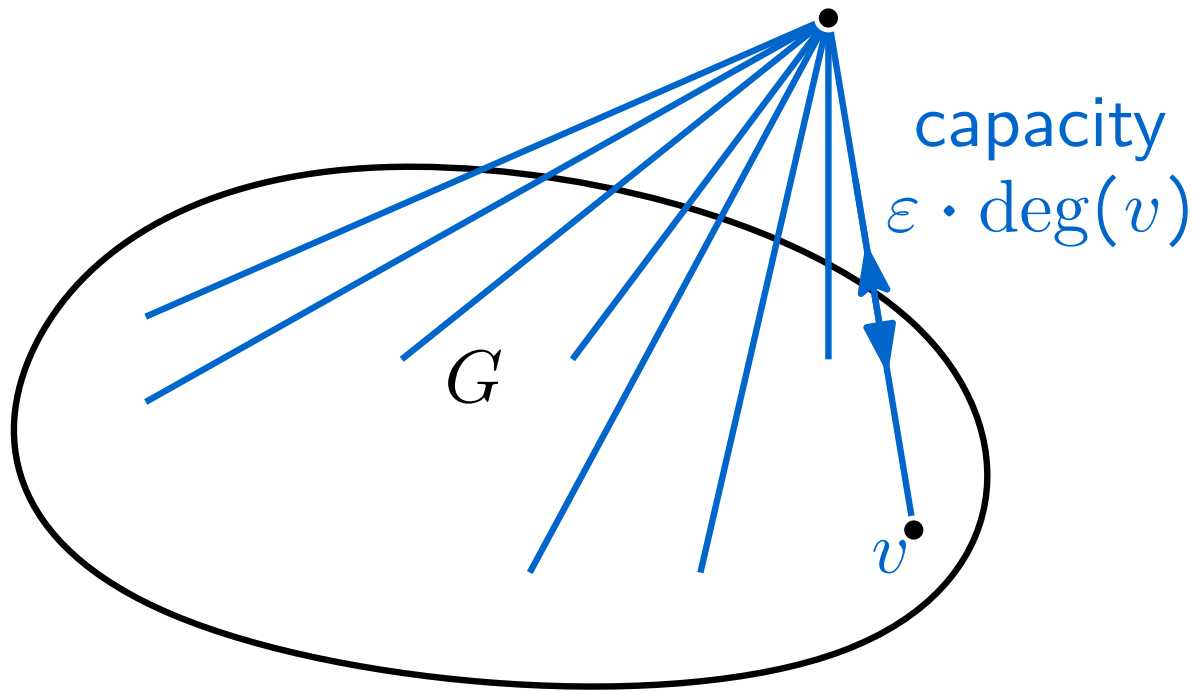
Max-flow in $G + \text{star} \neq \text{Max-Flow in } G$

New Idea: Shortcuts

Guarantee: Suppose max-flow f^* of G is short

How to guarantee this?

Answer: We will cheat! “Shortcut Star”



Guarantee satisfied in $G + \text{star}$

Compute max flow in $G + \text{star}$ in $\tilde{O}(m/\epsilon)$ time!

Big Problem:

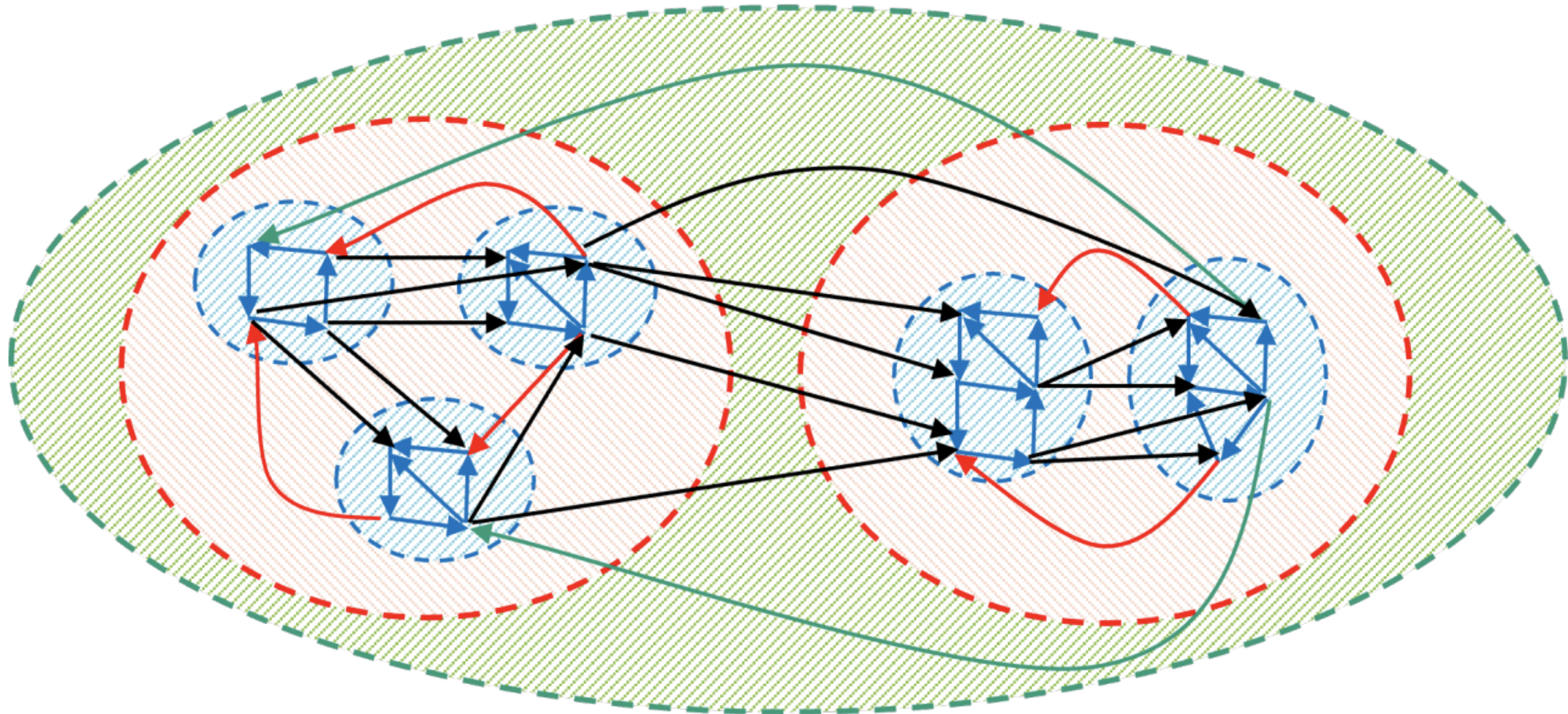
Max-flow in $G + \text{star} \neq \text{Max-Flow in } G$

Luckily:

Max-flow in $G + \text{star} \approx \text{Max-Flow in } G$
if G is ϕ -expander and ϵ is small

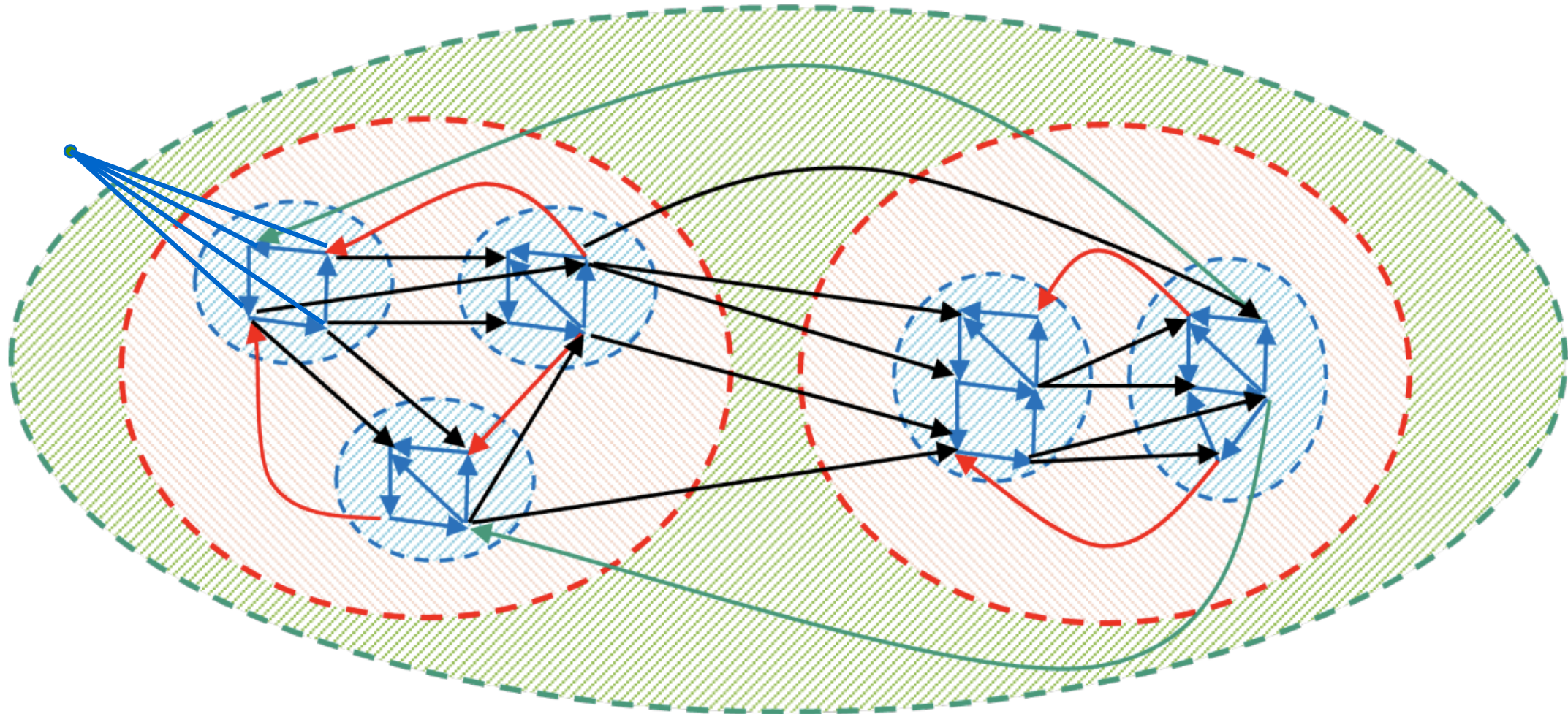
Shortcuts + Expander Hierarchy

Add shortcut stars on each expander in the Expander Hierarchy



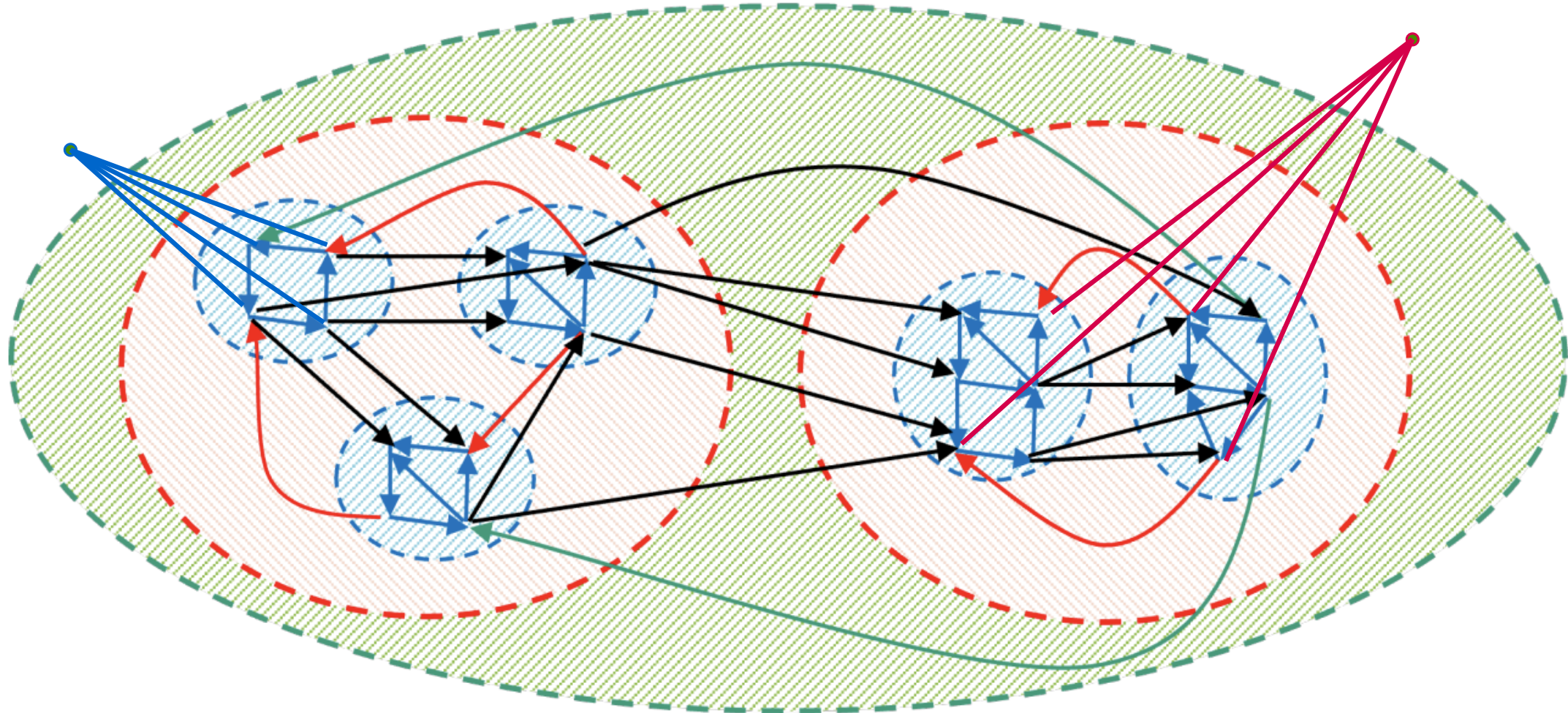
Shortcuts + Expander Hierarchy

Add shortcut stars on each expander in the Expander Hierarchy



Shortcuts + Expander Hierarchy

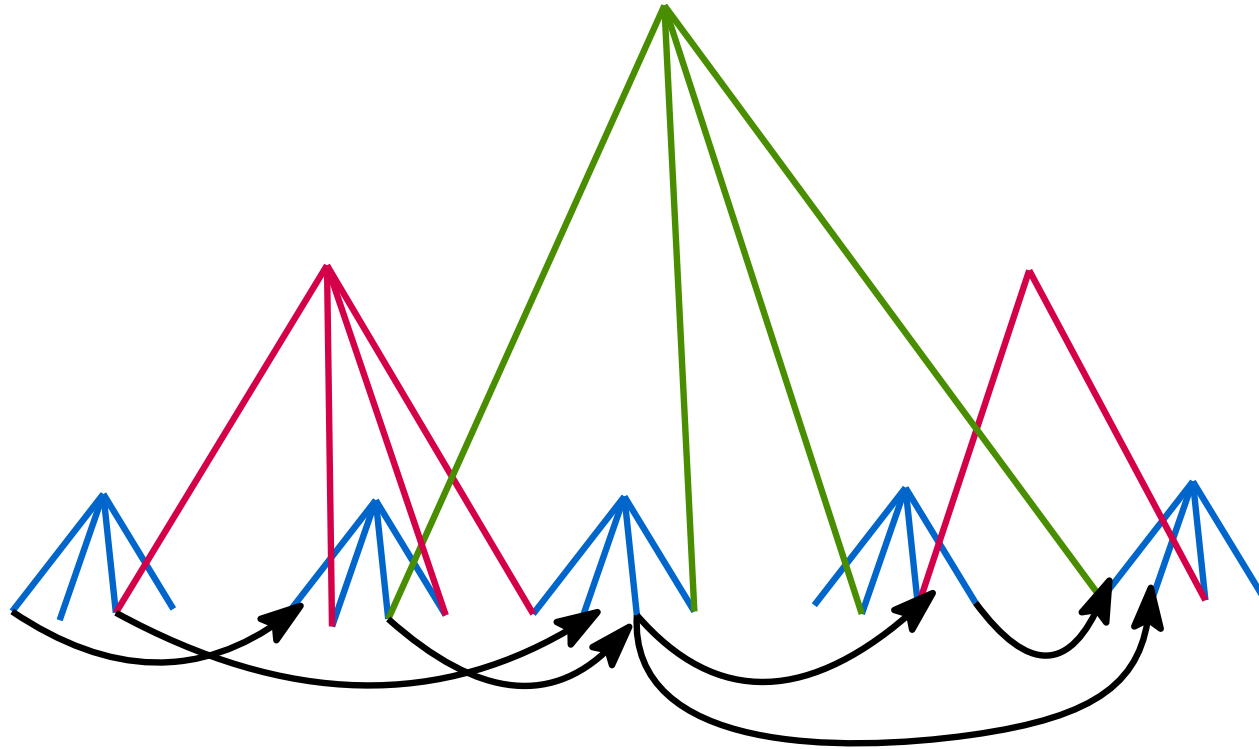
Add shortcut stars on each expander in the Expander Hierarchy



Shortcuts + Expander Hierarchy

Add shortcut stars on each expander in the Expander Hierarchy

Much simpler structure \approx DAG + $\log n$ levels of disjoint stars



Shortcuts + Expander Hierarchy

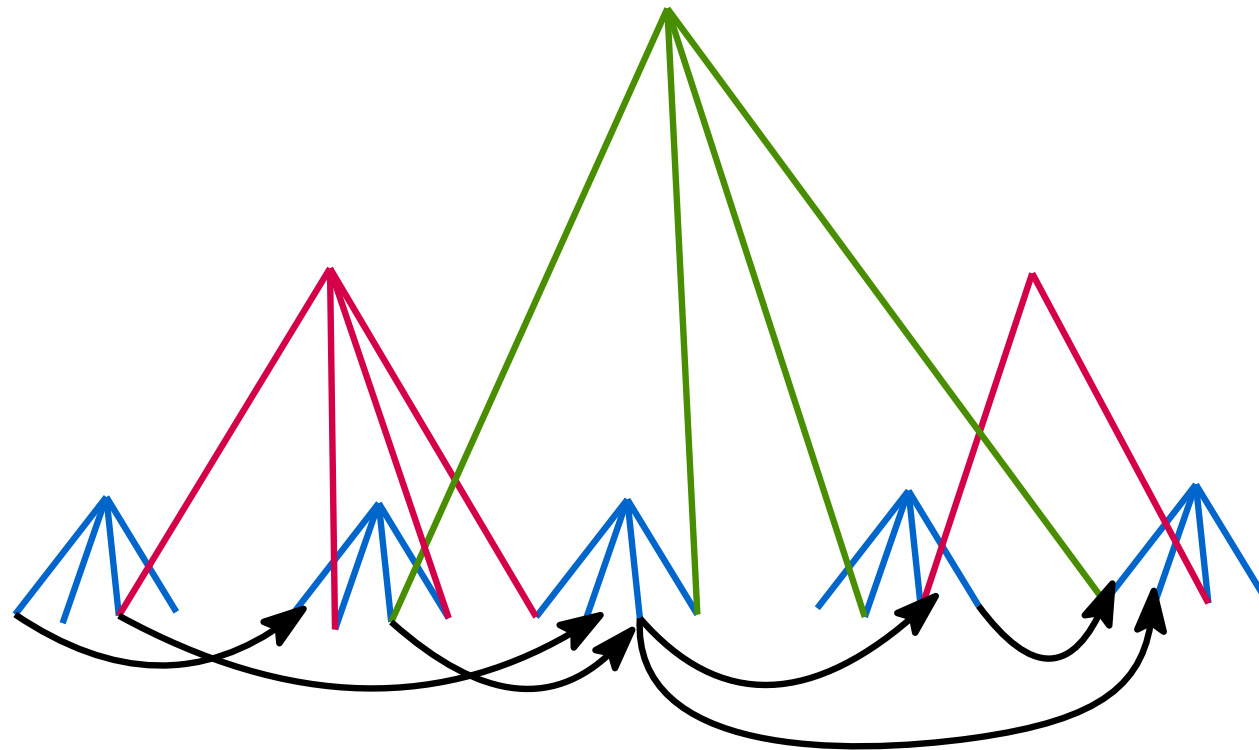
Add shortcut stars on each expander in the Expander Hierarchy

Much simpler structure \approx DAG + $\log n$ levels of disjoint stars

Greatly Simplifies:

Construction of hierarchy

Most of the remaining analysis



Shortcuts + Expander Hierarchy

Add shortcut stars on each expander in the Expander Hierarchy

Much simpler structure \approx DAG + $\log n$ levels of disjoint stars

Greatly Simplifies:

Construction of hierarchy

Most of the remaining analysis

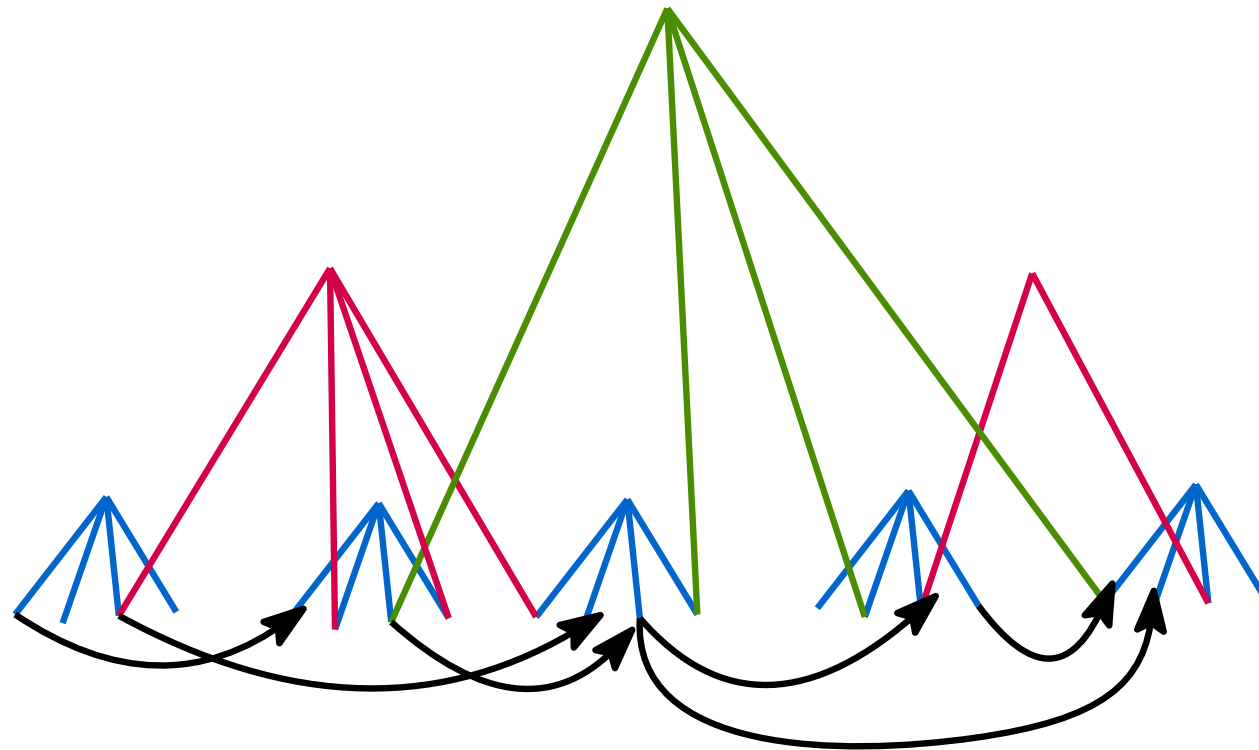
Consequences:

$$n^{2+o(1)} \mapsto \tilde{O}(n^2)$$

40 instead of 100 pages

Implementable

Deterministic for vertex-capacities



Summary

Main Result: Maximum flow in on n -vertex graphs in $O(n^2 \log^{19} n)$ time.

Theoretically possible \mapsto Implementable \mapsto Practical

Techniques:

Augmenting Paths (new version of Push-Relabel)

Directed Expander Hierarchy

New: Shortcuts

My Hope:

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

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

Bottleneck towards $\tilde{O}(m)$:
Approximate Max Flow in DAGs
(“simple” $\tilde{O}(n^2)$ algo)

Summary

Main Result: Maximum flow in on n -vertex graphs in $O(n^2 \log^{19} n)$ time.

Theoretically possible \mapsto Implementable \mapsto Practical

Techniques:

Augmenting Paths (new version of Push-Relabel)

Directed Expander Hierarchy

New: Shortcuts

My Hope:

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

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

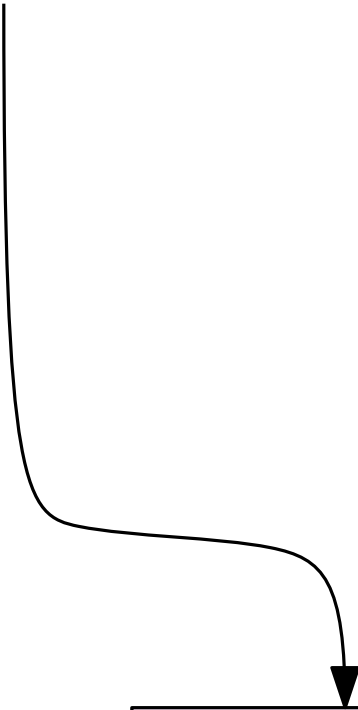
Thanks!

Extra Slides

Outline

1. Recap: Push-Relabel

Graph G



Push Relabel

Max Flow

Outline

1. Recap: Push-Relabel
2. Weighted PR

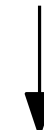
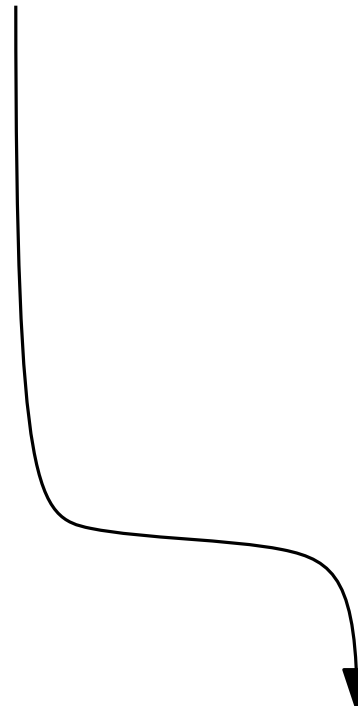
Graph G

“Good” edge lengths

“hint”

Weighted Push Relabel

Approximate Max Flow



Outline

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

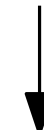
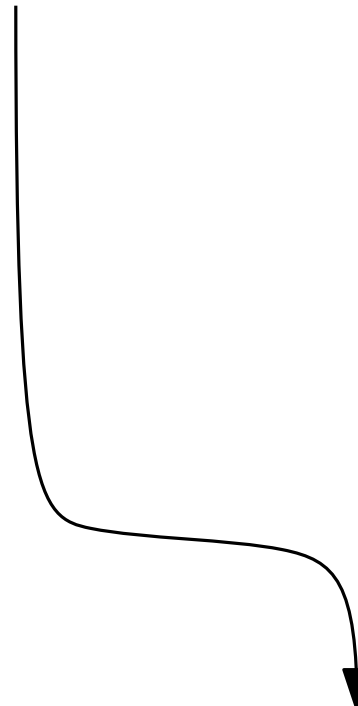
Graph G

“Good” edge lengths

“hint”

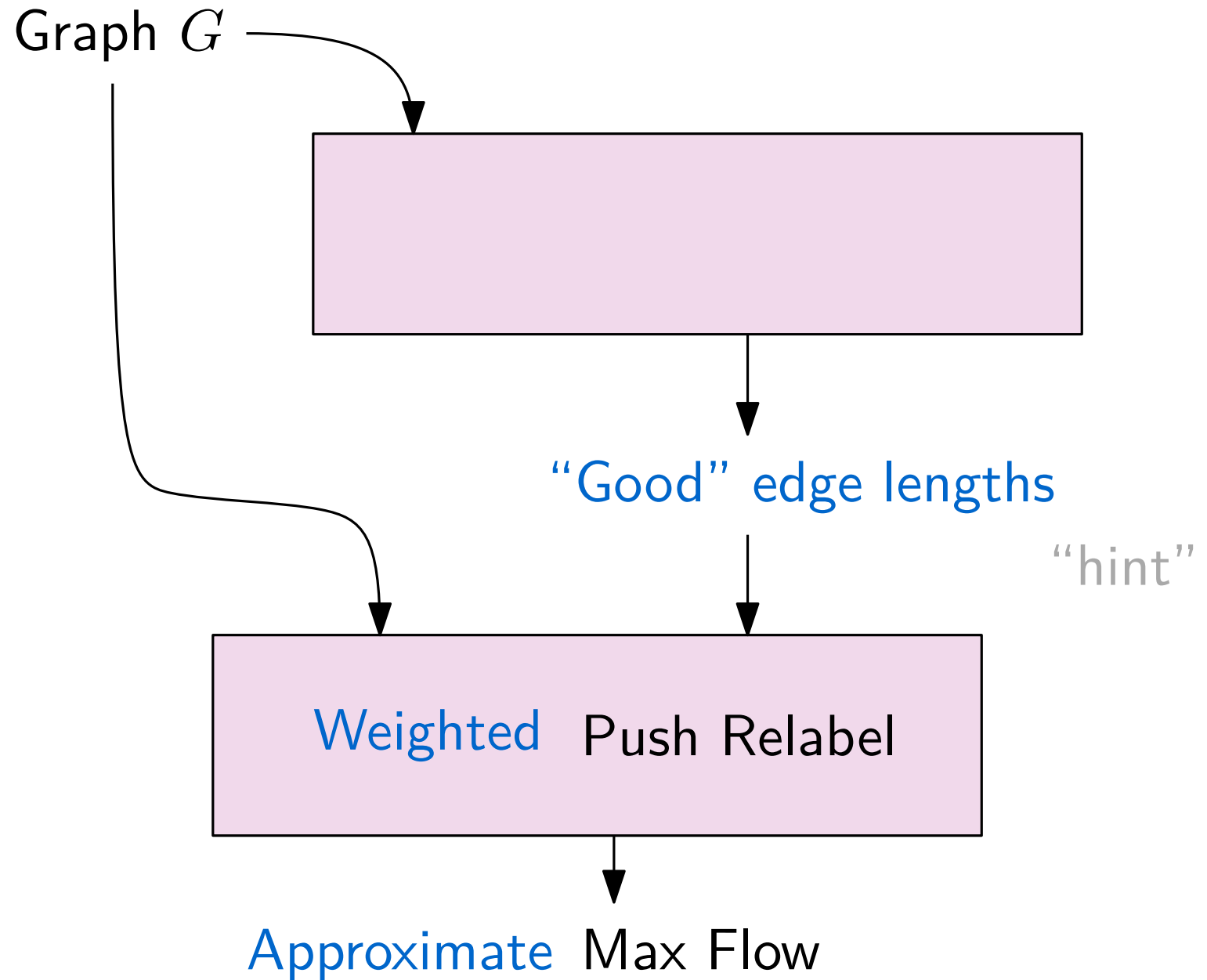
Weighted Push Relabel

Approximate Max Flow



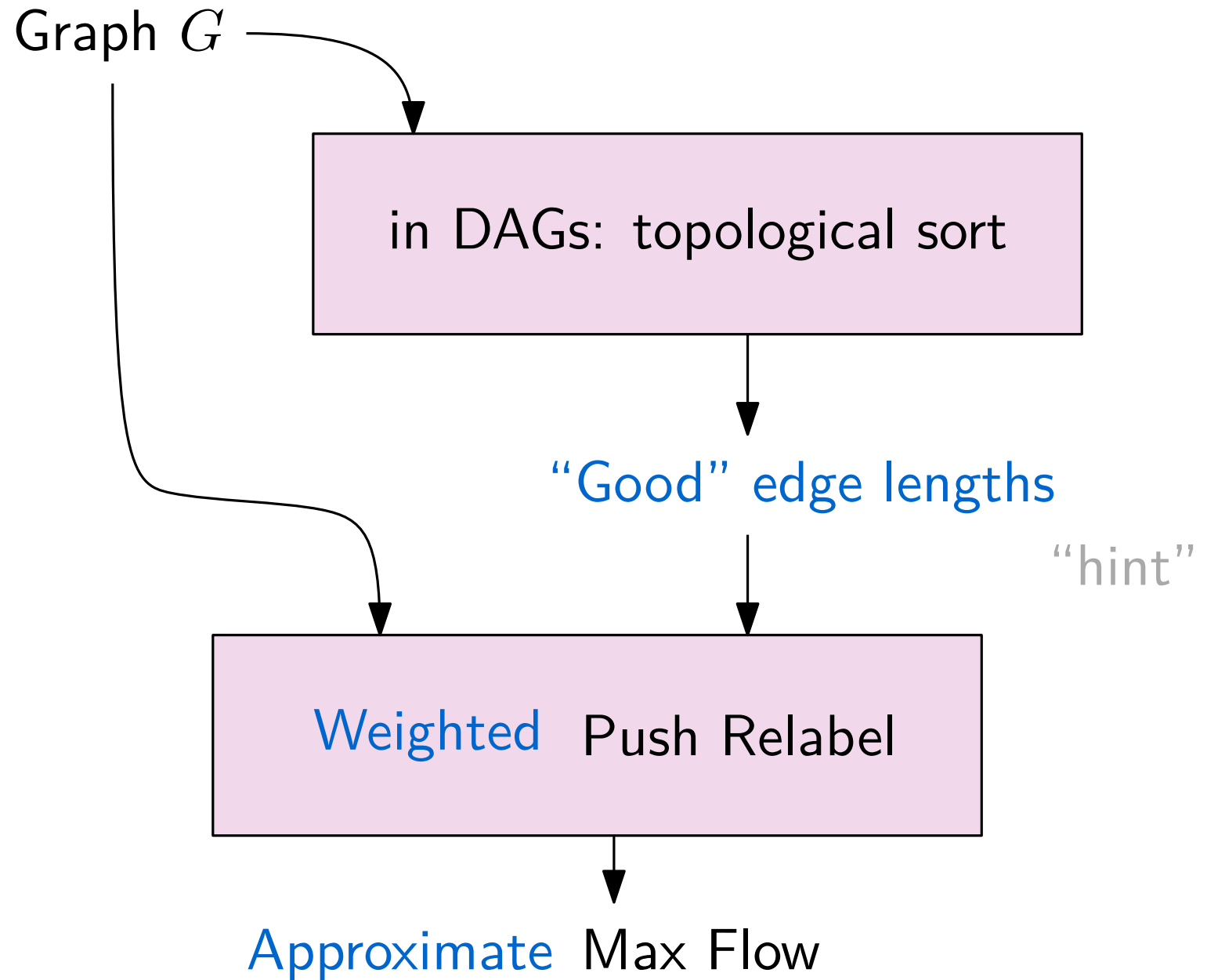
Outline

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



Outline

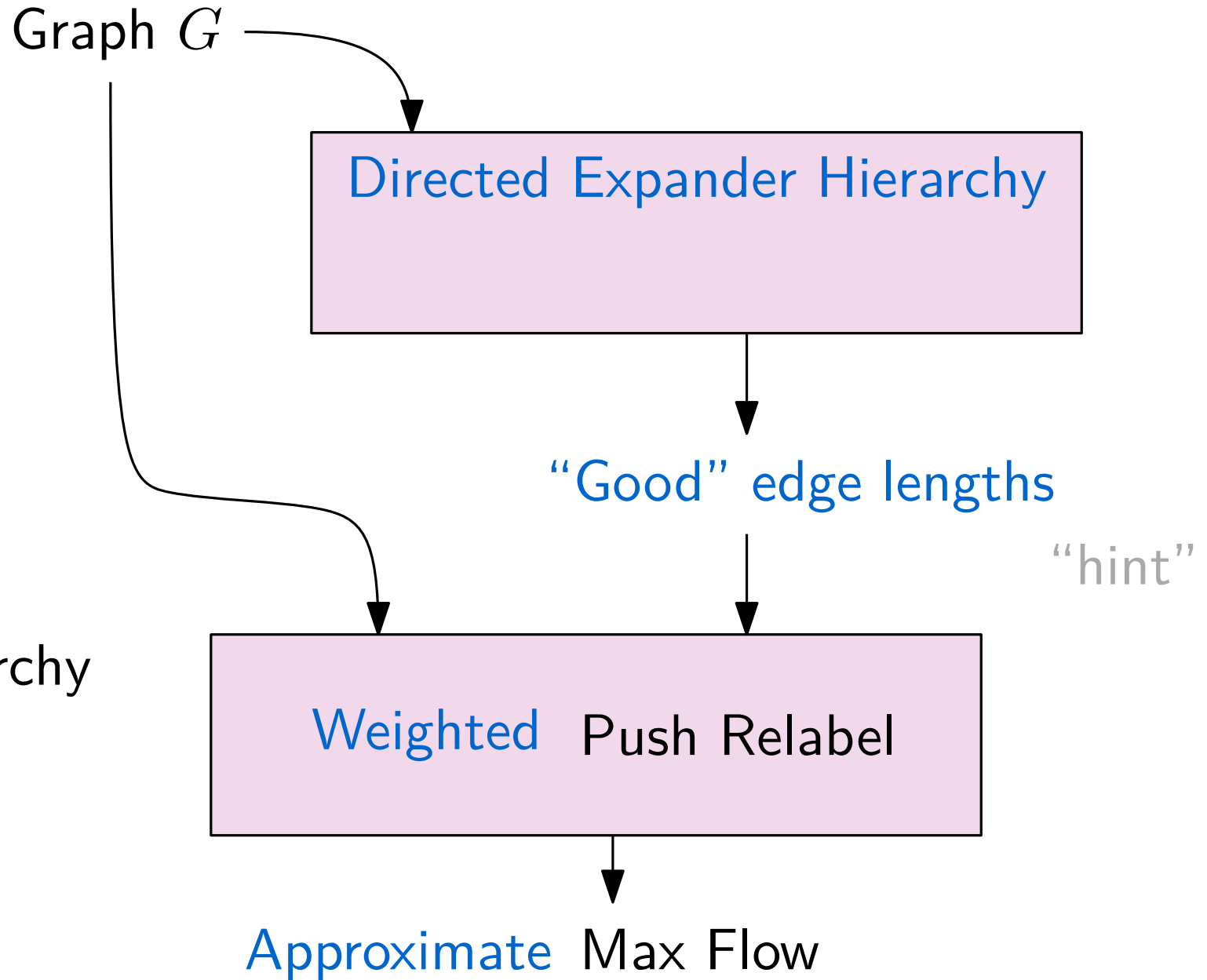
1. Recap: Push-Relabel
2. Weighted PR
3. ~~Good~~
4. Edge Lengths in DAGs



Outline

1. Recap: Push-Relabel
2. Weighted PR
3. “Good”
4. Edge Lengths in DAGs
5. General Graphs:

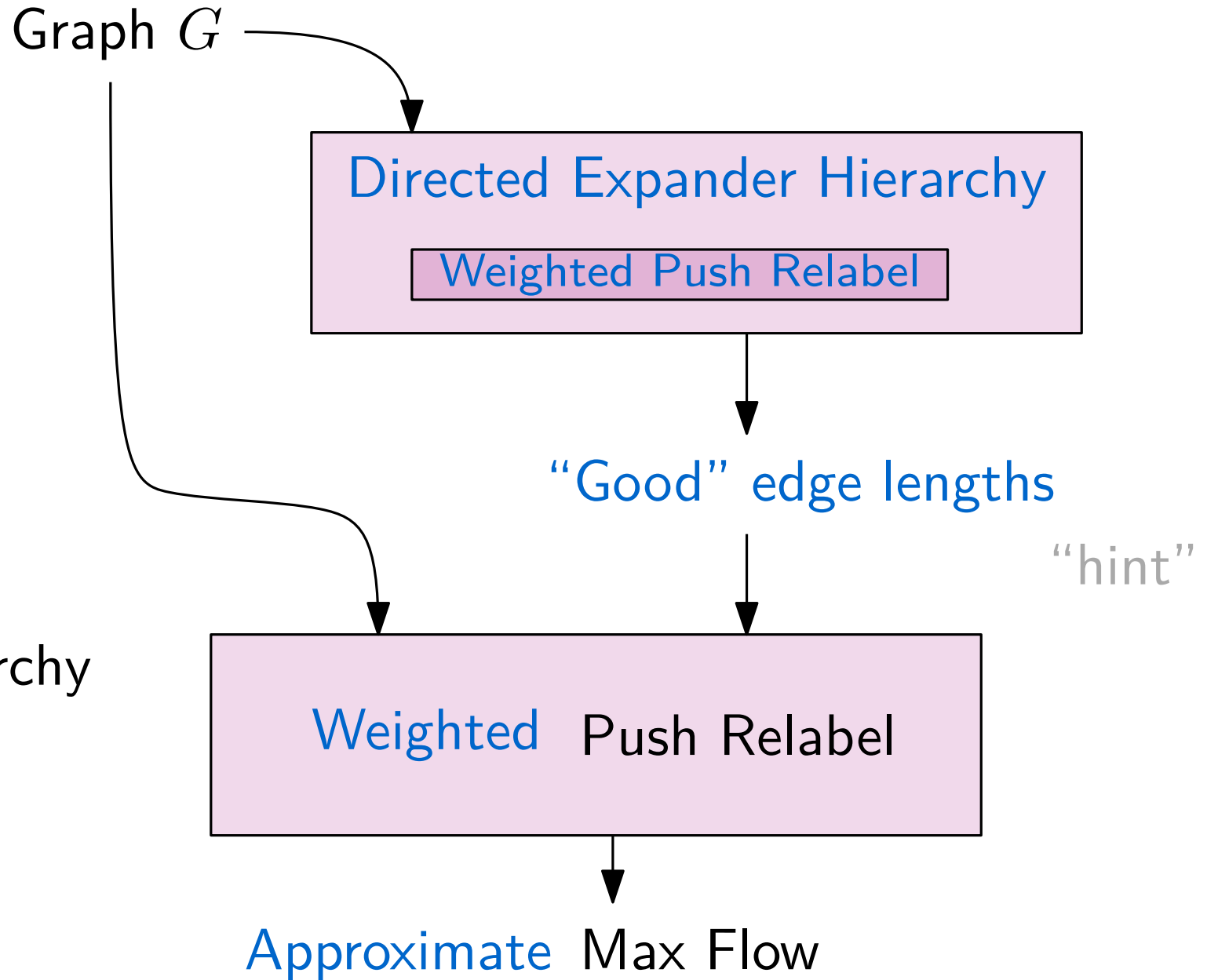
Directed Expander Hierarchy



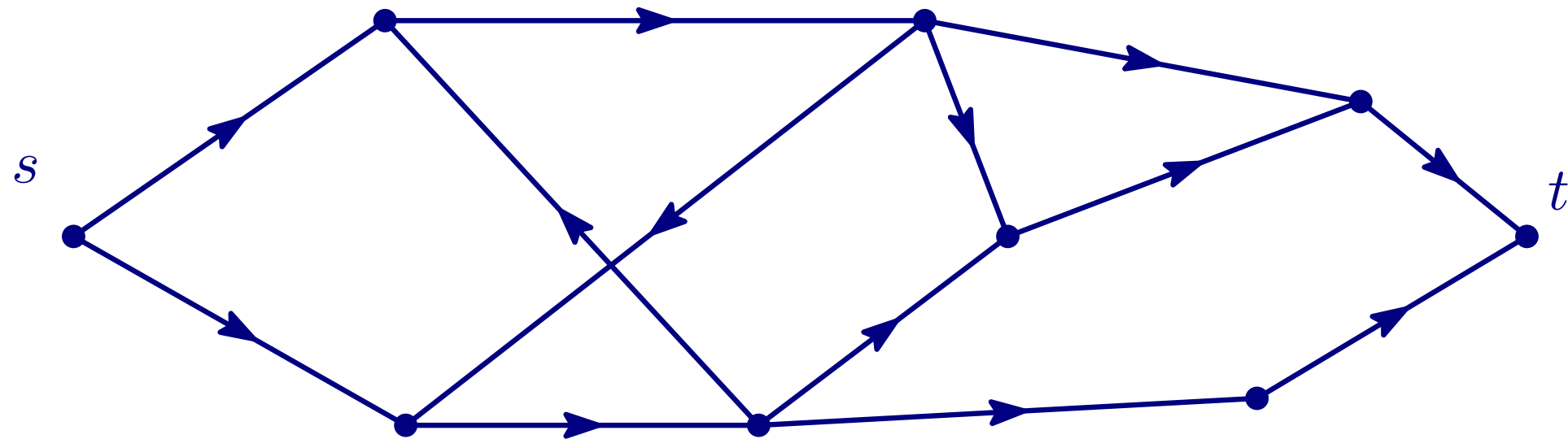
Outline

1. Recap: Push-Relabel
2. Weighted PR
3. “Good”
4. Edge Lengths in DAGs
5. General Graphs:

Directed Expander Hierarchy

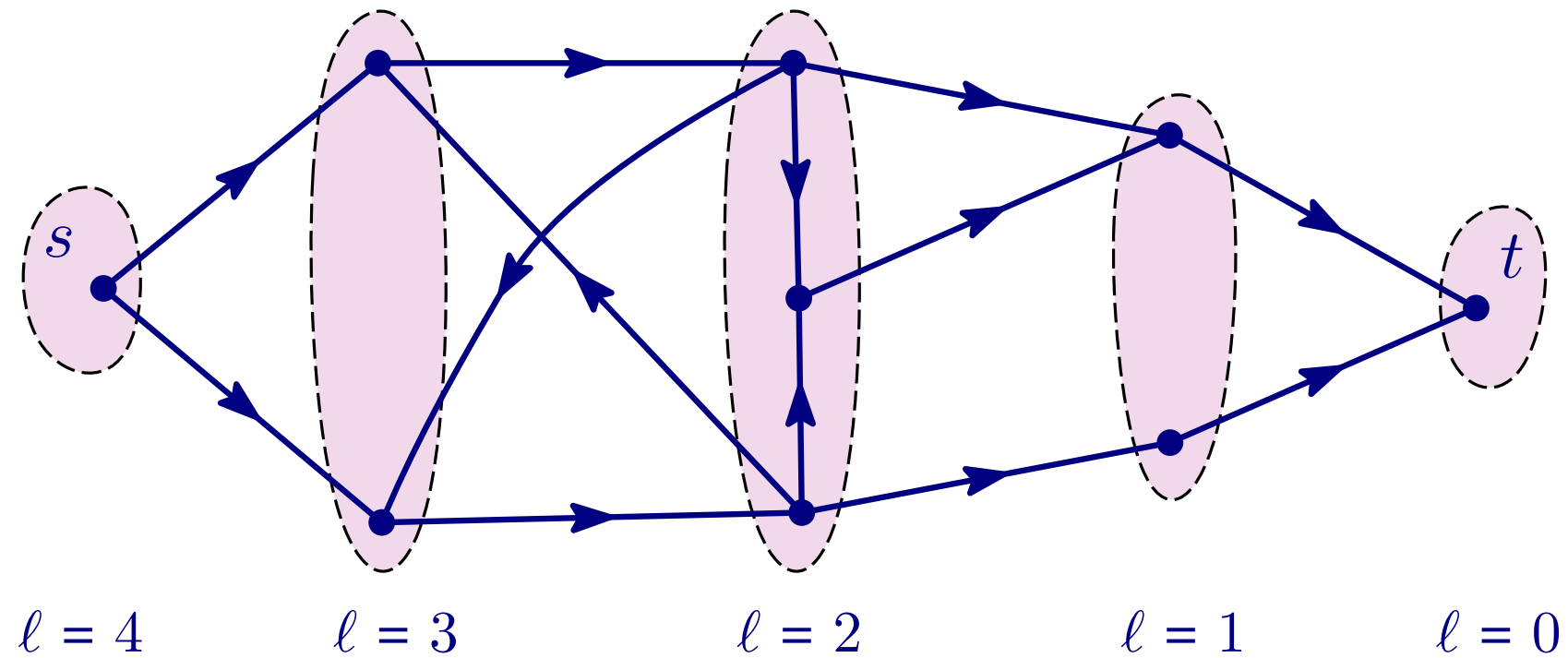


Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88]



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

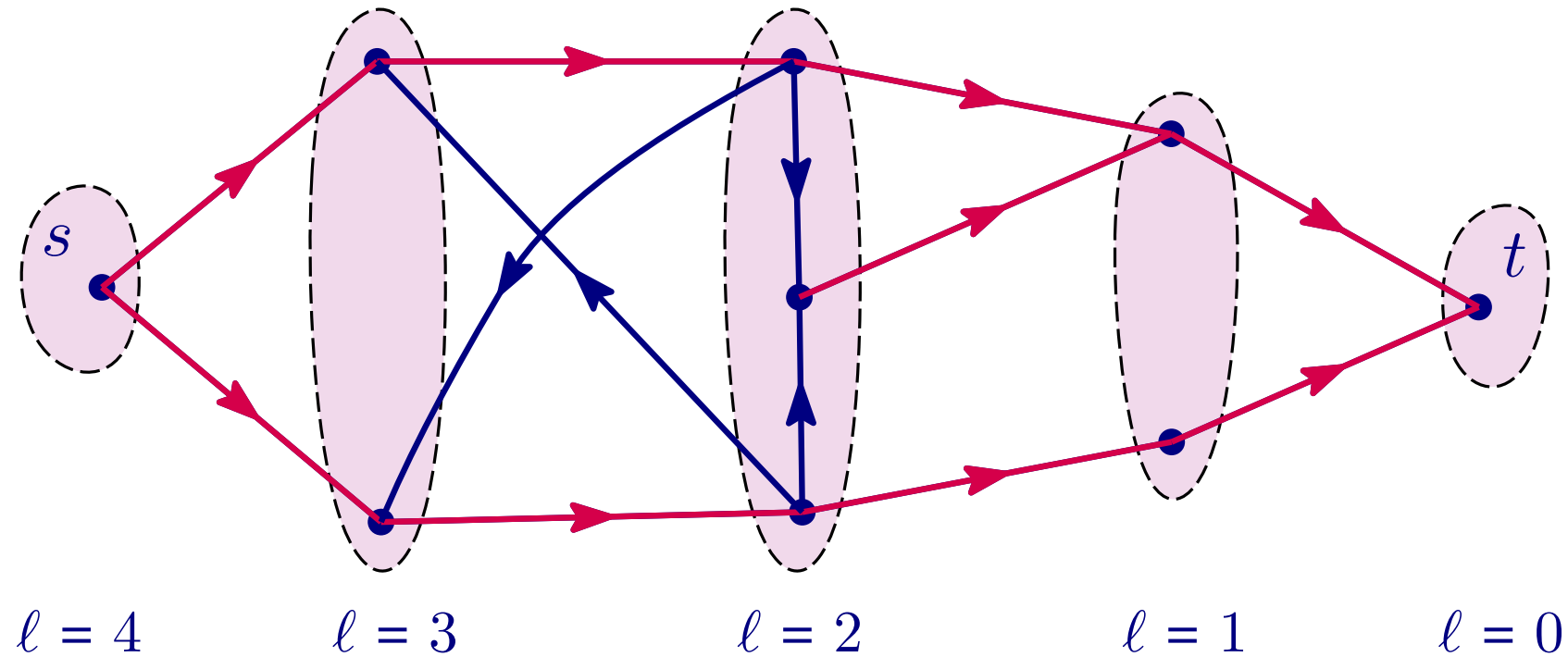
$$\ell(v) = \text{dist}(v, t)$$



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

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

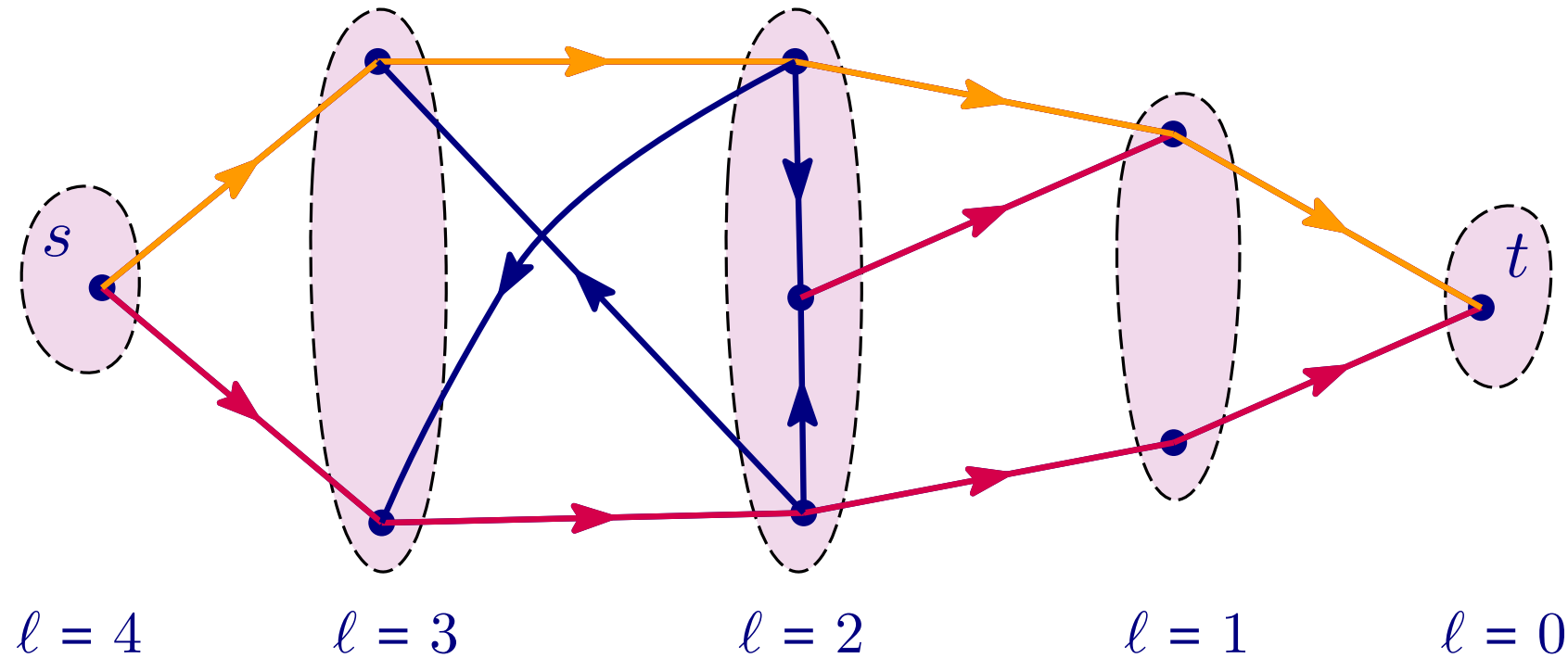
edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + 1$



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

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

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



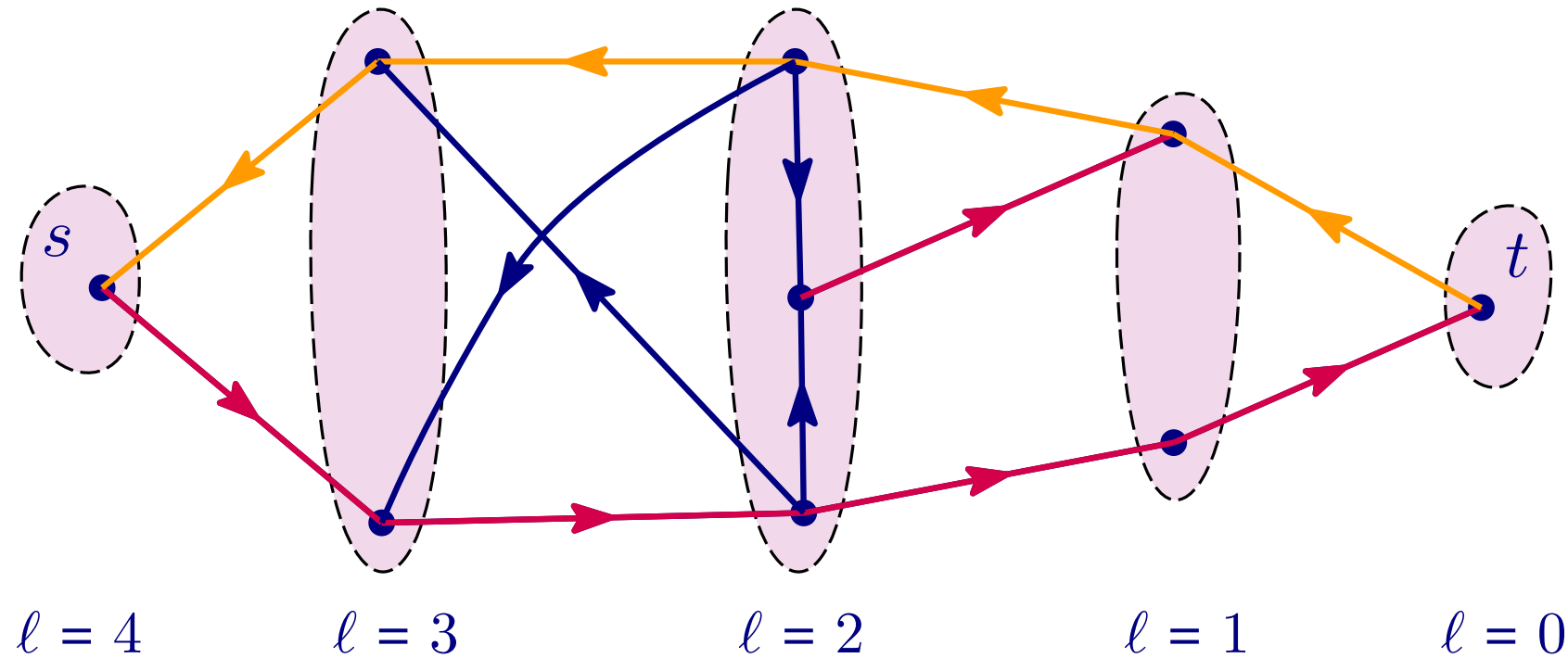
Shortest Augmenting Path: follow admissible edges from s

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

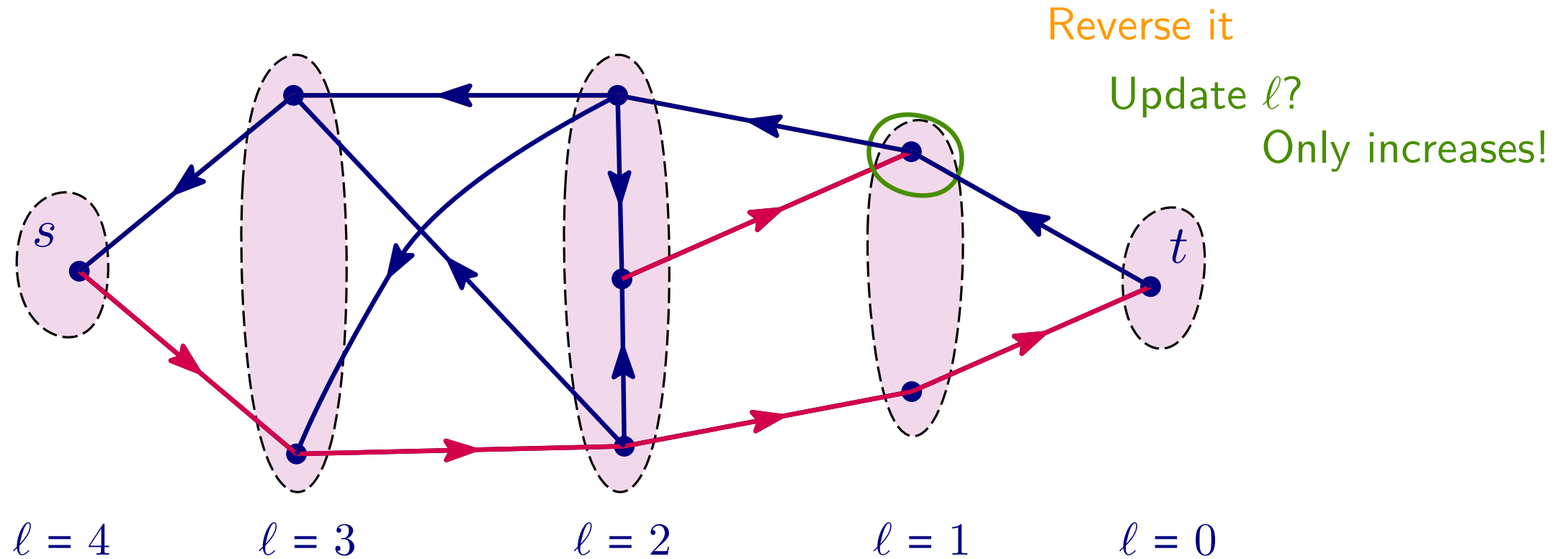


Shortest Augmenting Path: follow admissible edges from s

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

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

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

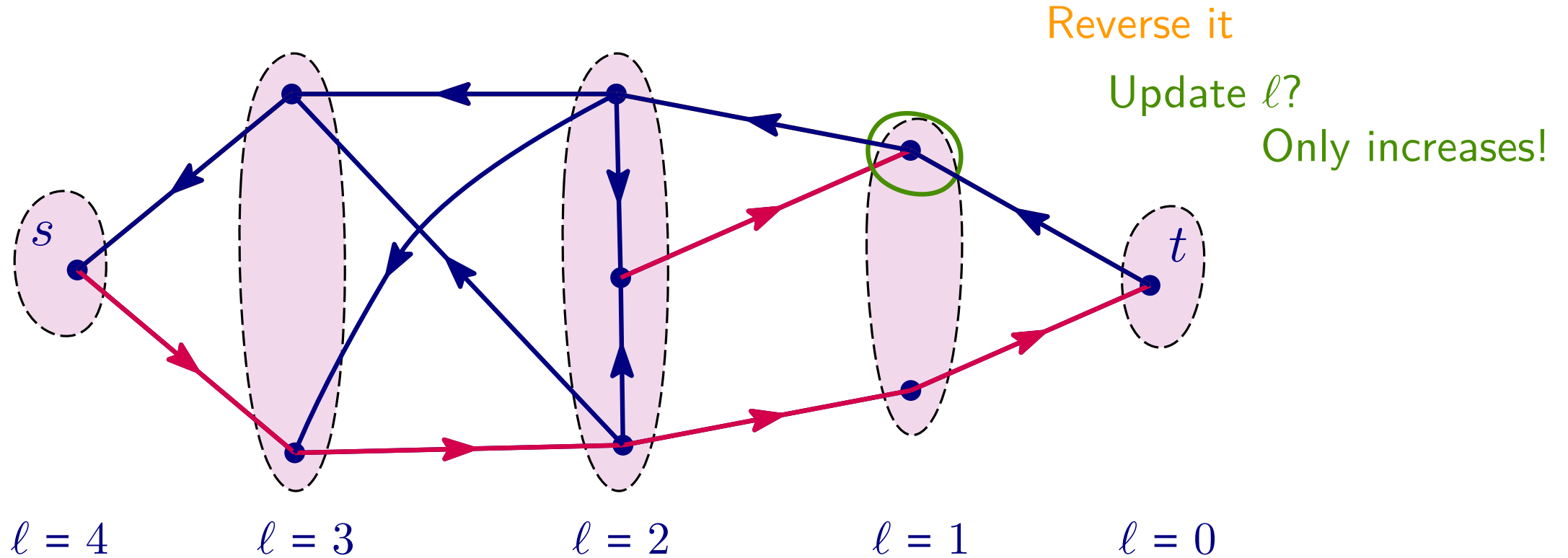


Shortest Augmenting Path: follow admissible edges from s

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

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

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



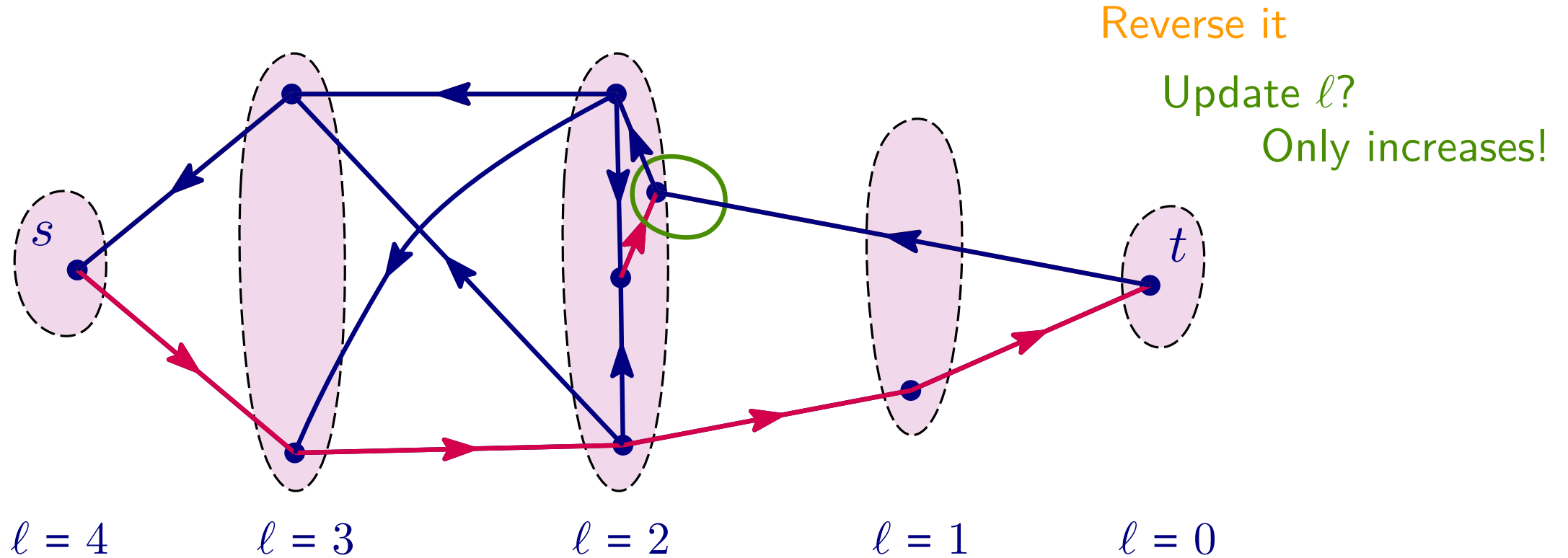
Shortest Augmenting Path: follow admissible edges from s

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

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

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

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



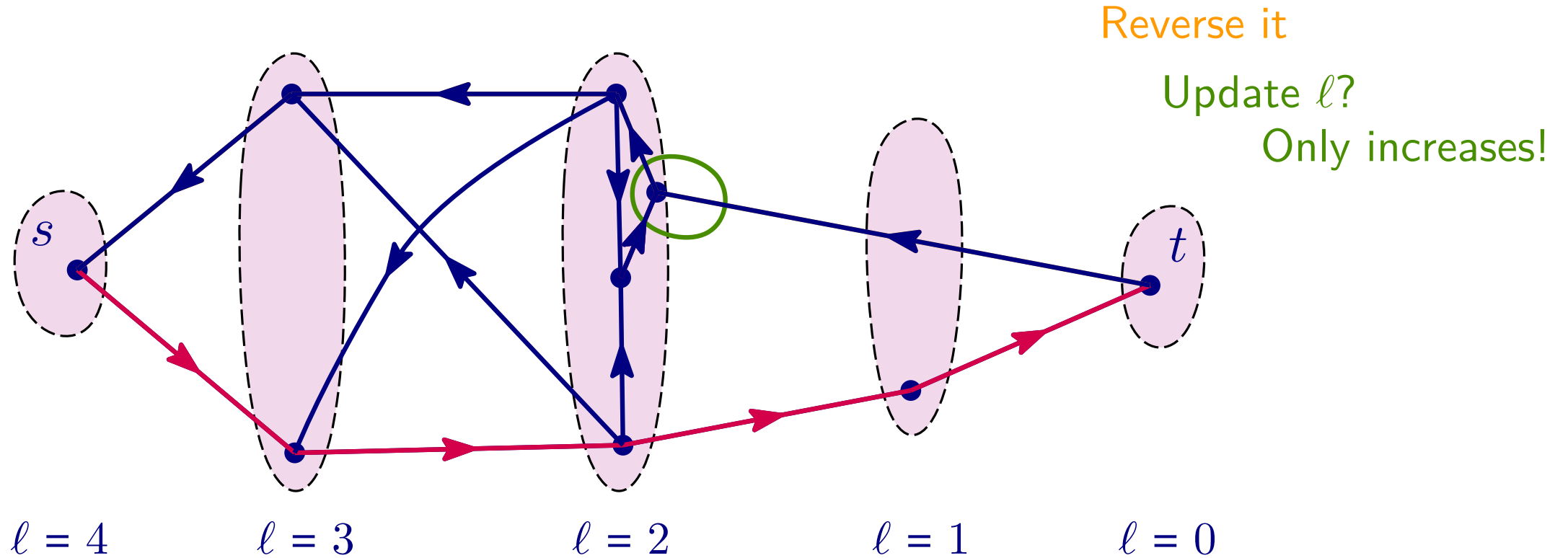
Shortest Augmenting Path: follow admissible edges from s

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

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

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

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



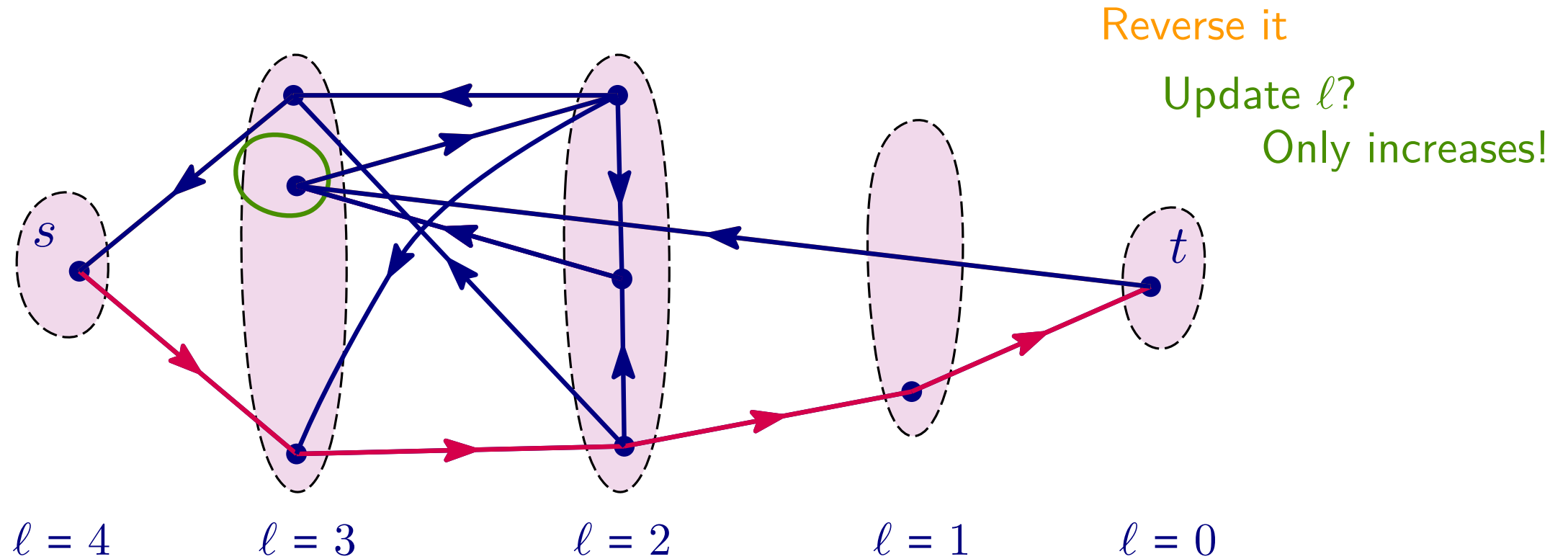
Shortest Augmenting Path: follow admissible edges from s

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

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

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

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



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]

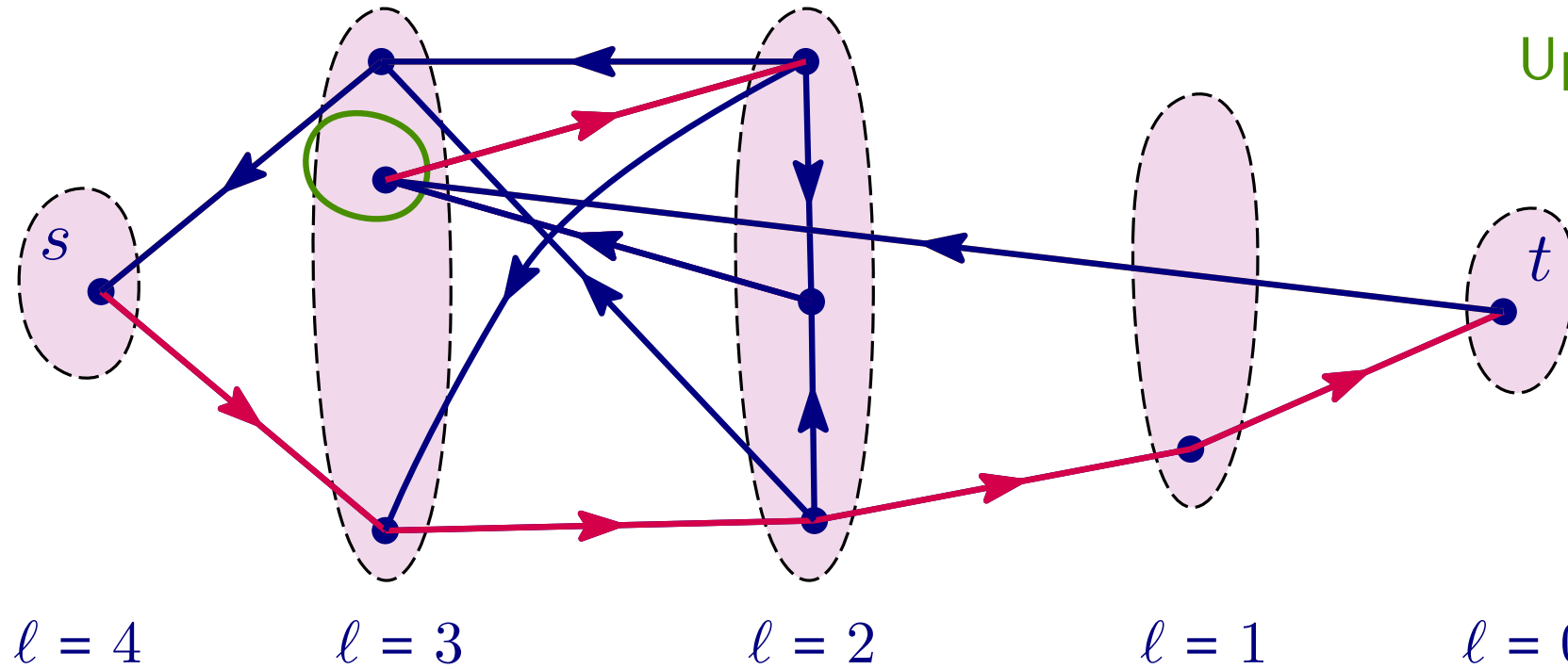
$$\ell(v) = \text{dist}(v, t)$$

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

$$\text{RELABEL}(v)$$

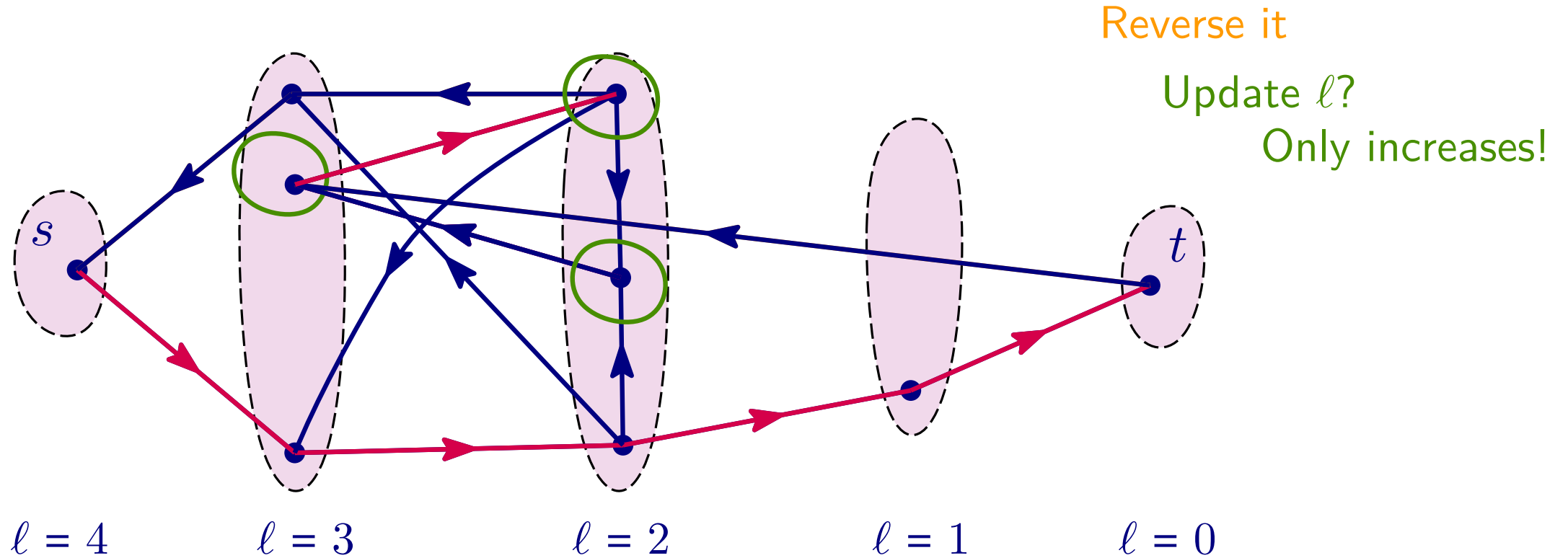
If no **admissible** out-edge:

$$\ell(v) \leftarrow \ell(v) + 1$$

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

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

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



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]

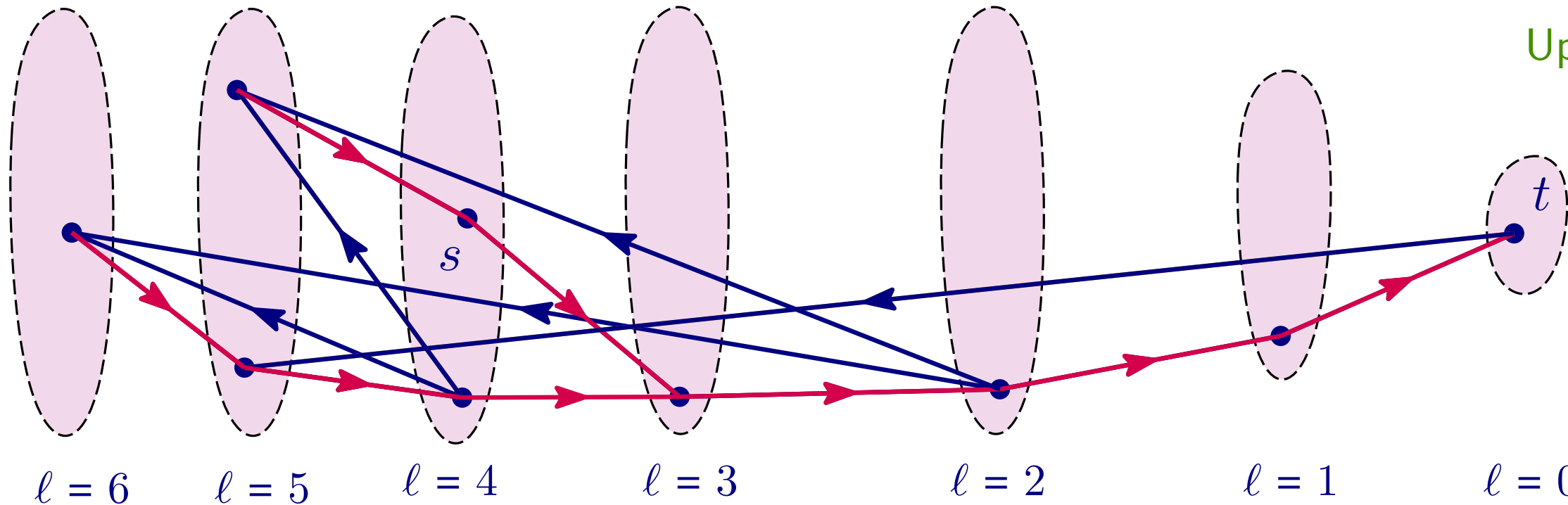
$$\ell(v) = \text{dist}(v, t)$$

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)

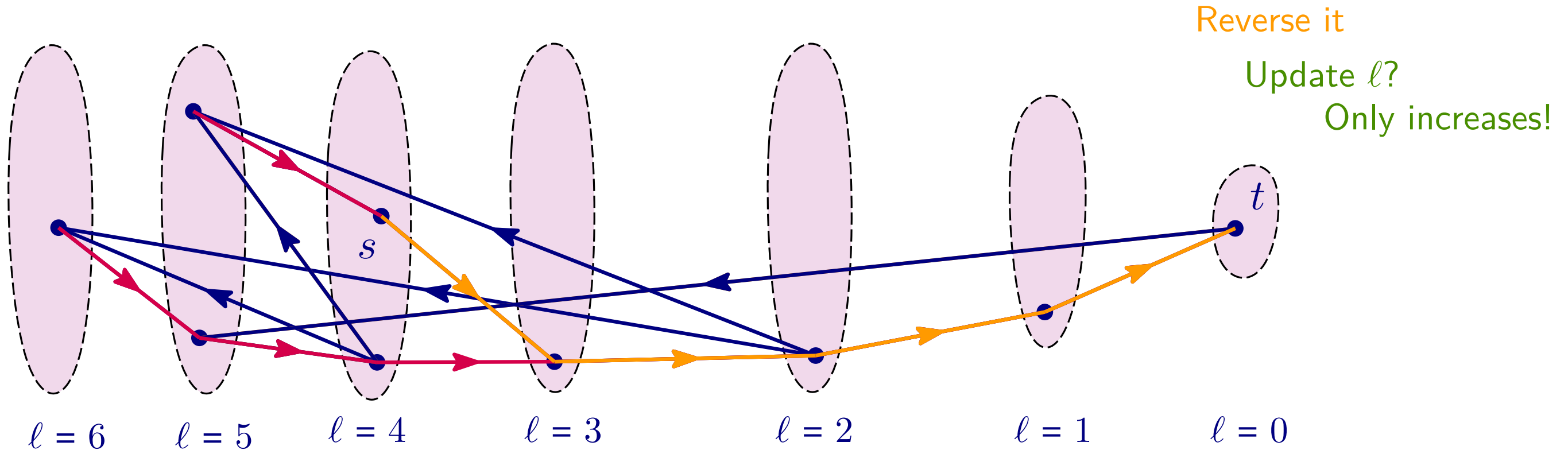
If no **admissible** out-edge:

$$\ell(v) \leftarrow \ell(v) + 1$$

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

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

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



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

$O(n^2)$ (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)



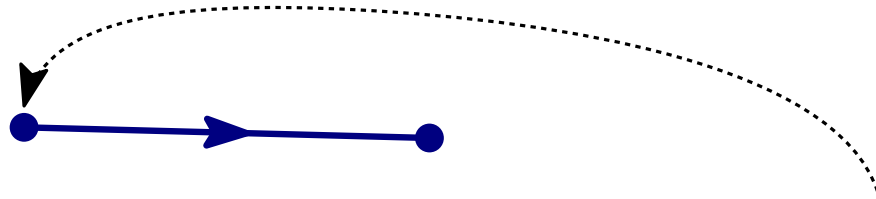
Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88] —Analysis

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)



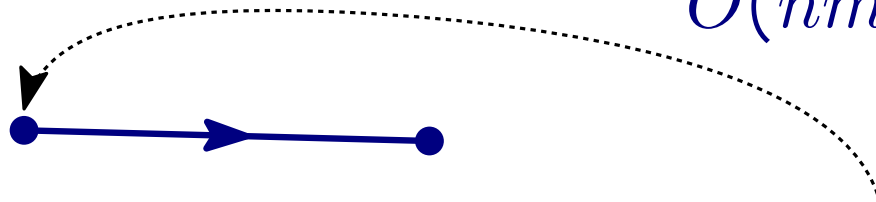
Push-Relabel / Augment-Relabel [Goldberg-Tarjan'88] —Analysis

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

Keeping track of admissible edges: $O(nm)$ (after relabel: recheck incident edges)

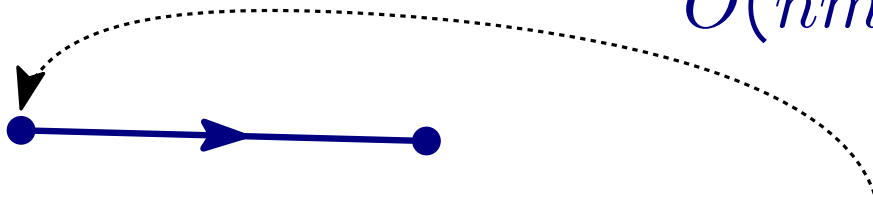
Augmentations $O(nm)$ (n per edge)

$O(nm \log n)$ (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




$$w(e) = 2$$




$$w(e) = 10$$

New Idea: Edge Lengths

“short” = “frequent”



$$w(e) = 2$$

“long” = “infrequent”



$$w(e) = 10$$

New Idea: Edge Lengths

“short” = “frequent”


$$w(e) = 2$$

“long” = “infrequent”


$$w(e) = 10$$


Guarantee: Path P in maxflow f^*




few long edges, potentially many short

New Idea: Edge Lengths

“short” = “frequent”


$$w(e) = 2$$

“long” = “infrequent”


$$w(e) = 10$$


Guarantee: Path P in maxflow f^\star
 $w(P) \leq \tilde{O}(n)$




few long edges, potentially many short

New Idea: Edge Lengths

“short” = “frequent”


$$w(e) = 2$$

“long” = “infrequent”


$$w(e) = 10$$

Potential Faster Algo:

look for short paths

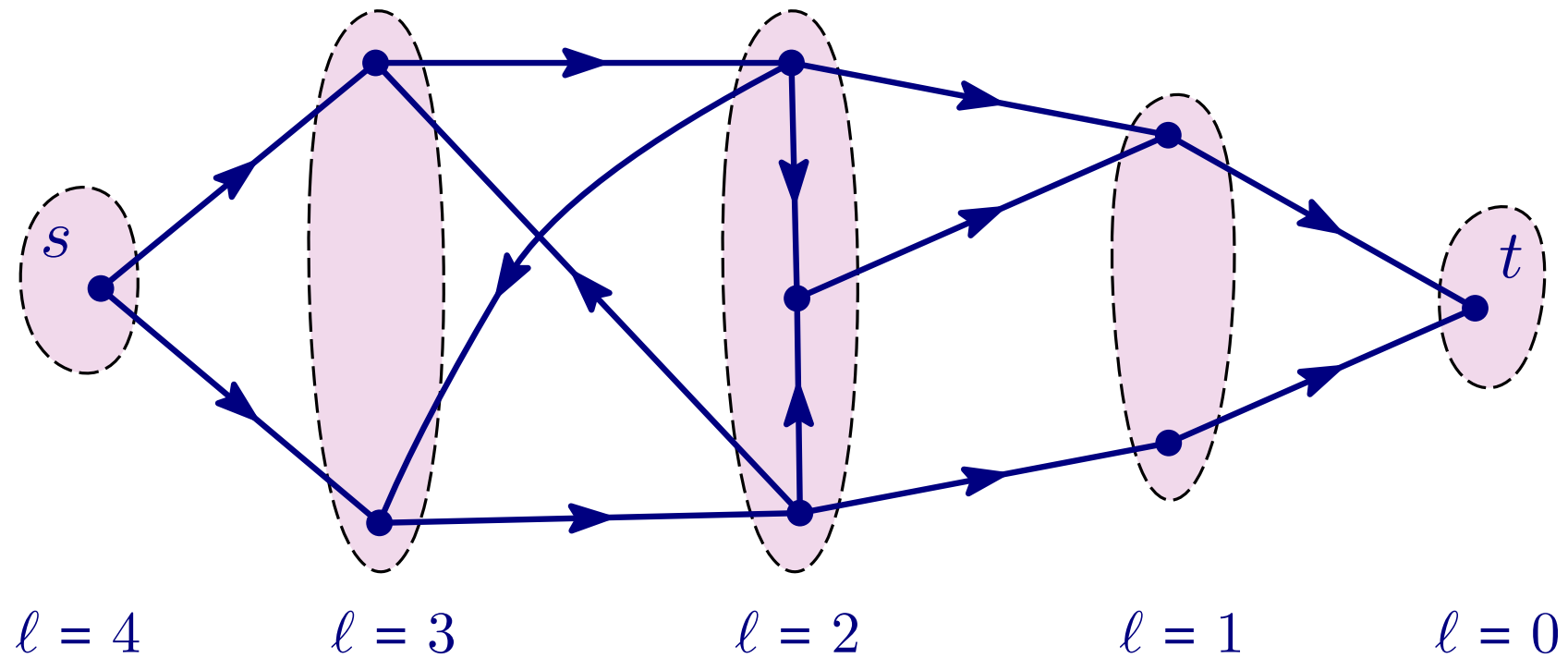
Guarantee: Path P in maxflow f^\star
 $w(P) \leq \tilde{O}(n)$



few long edges, potentially many short

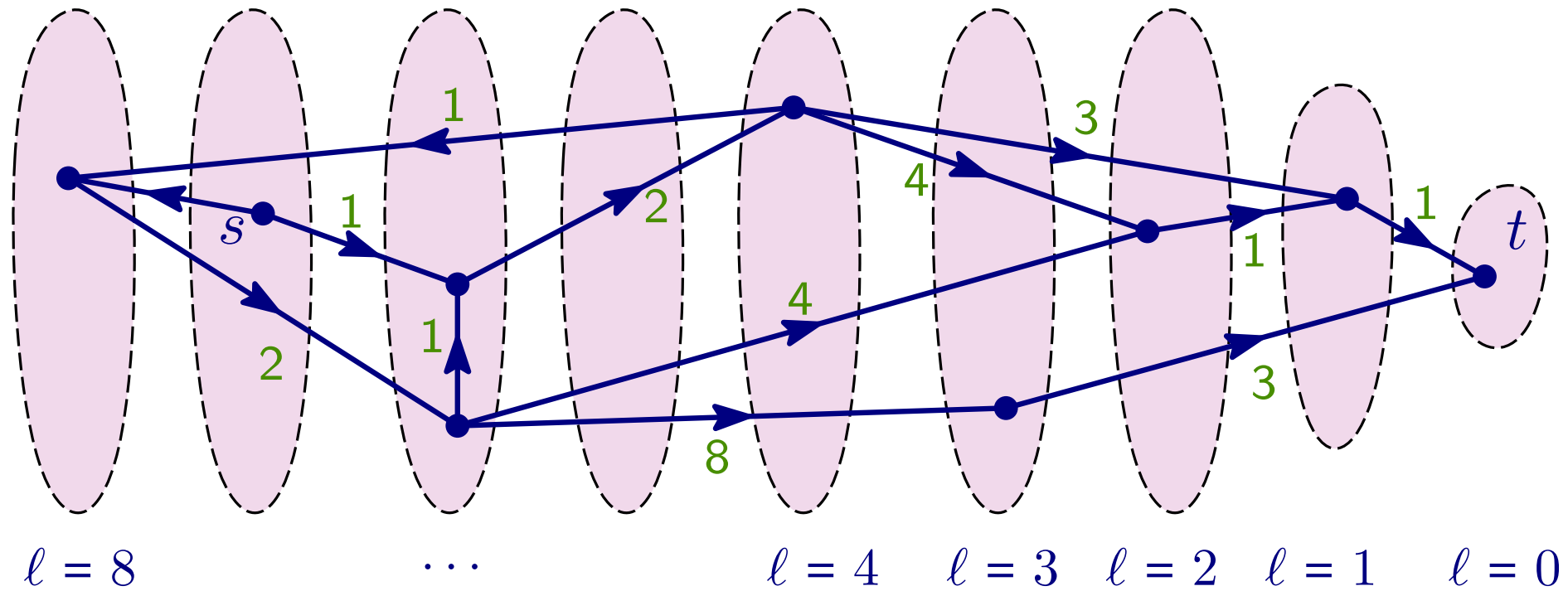
Weighted Push-Relabel

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



Weighted Push-Relabel

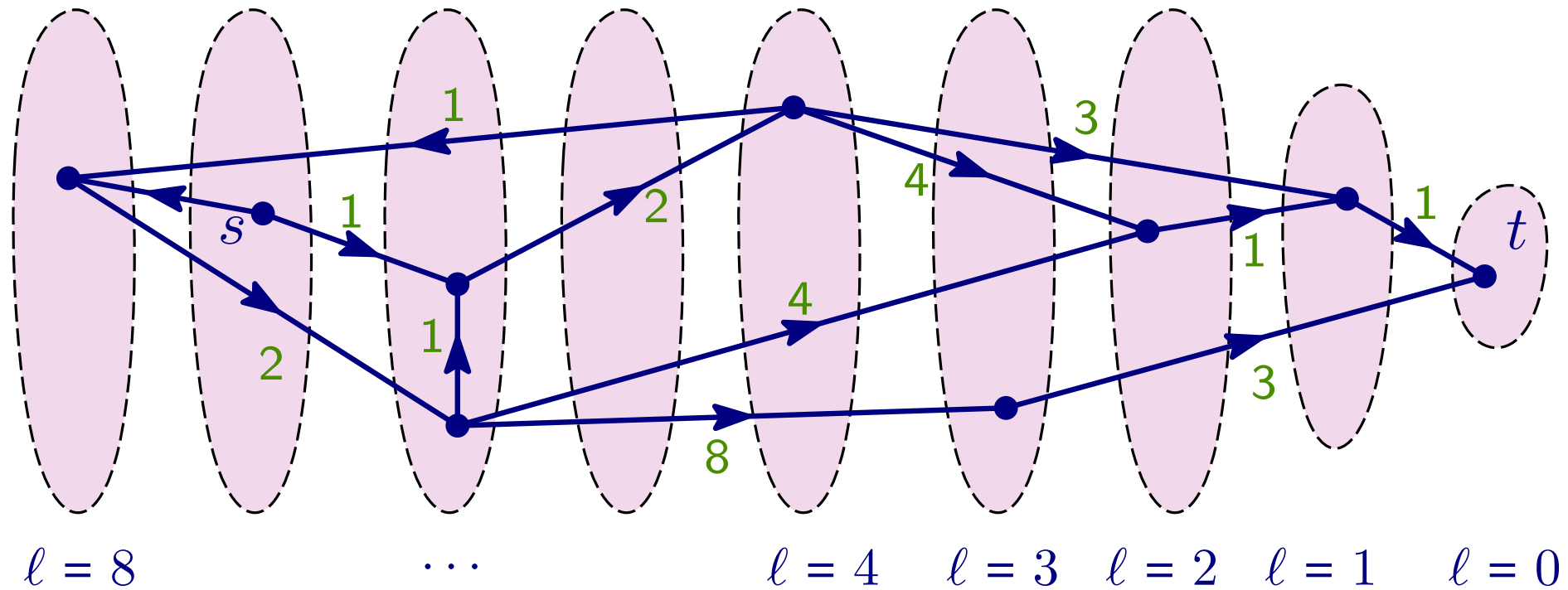
$$\ell(v) = \text{dist}_{\mathbf{w}}(v, t)$$



Weighted Push-Relabel

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

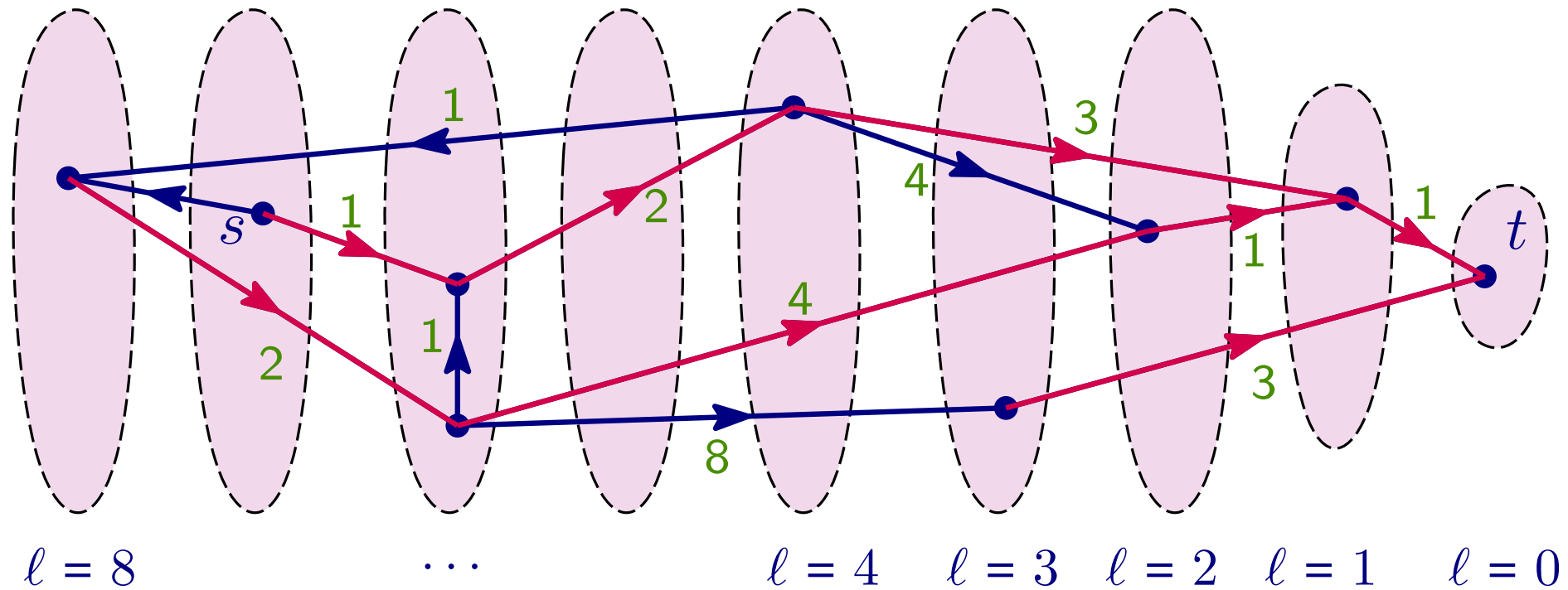
edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + 1$



Weighted Push-Relabel

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

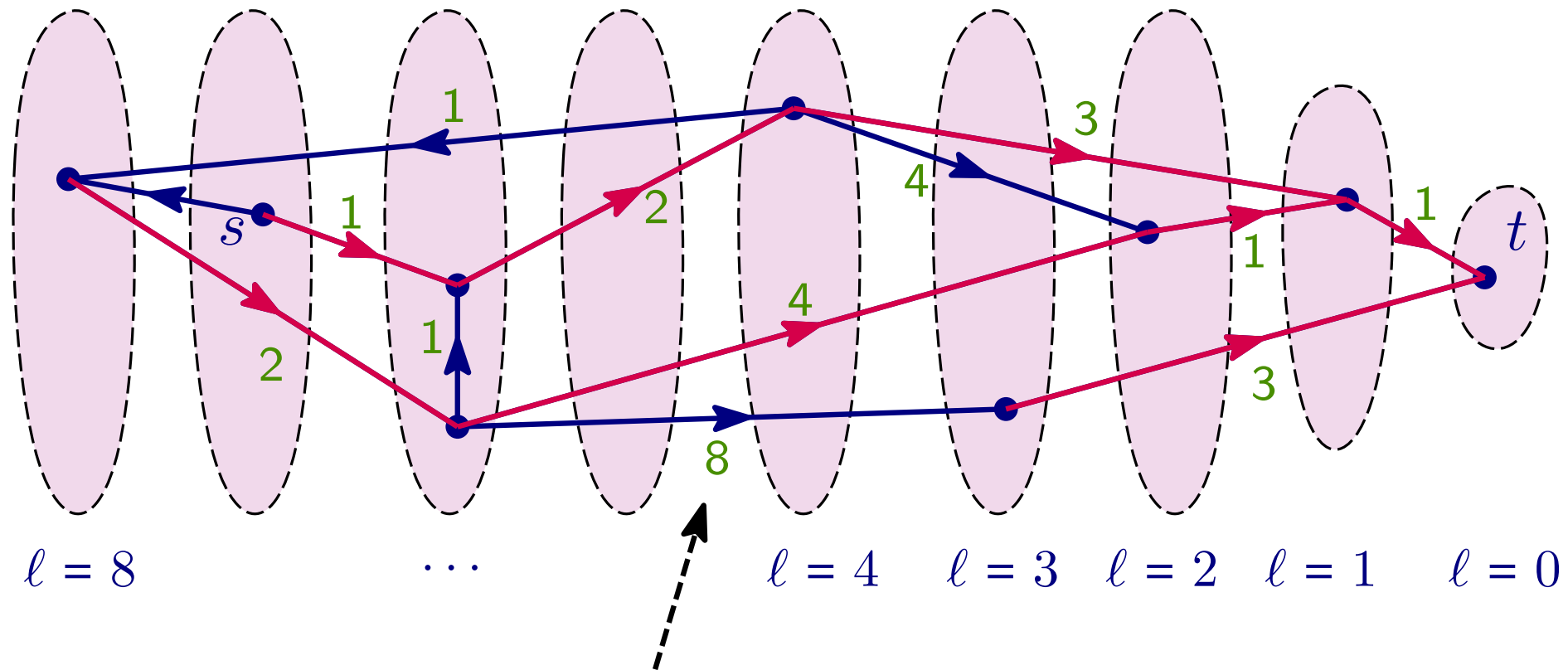
edge $e = (u, v)$ *admissible* iff $\ell(u) = \ell(v) + \mathbf{w}(e)$



Weighted Push-Relabel

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

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

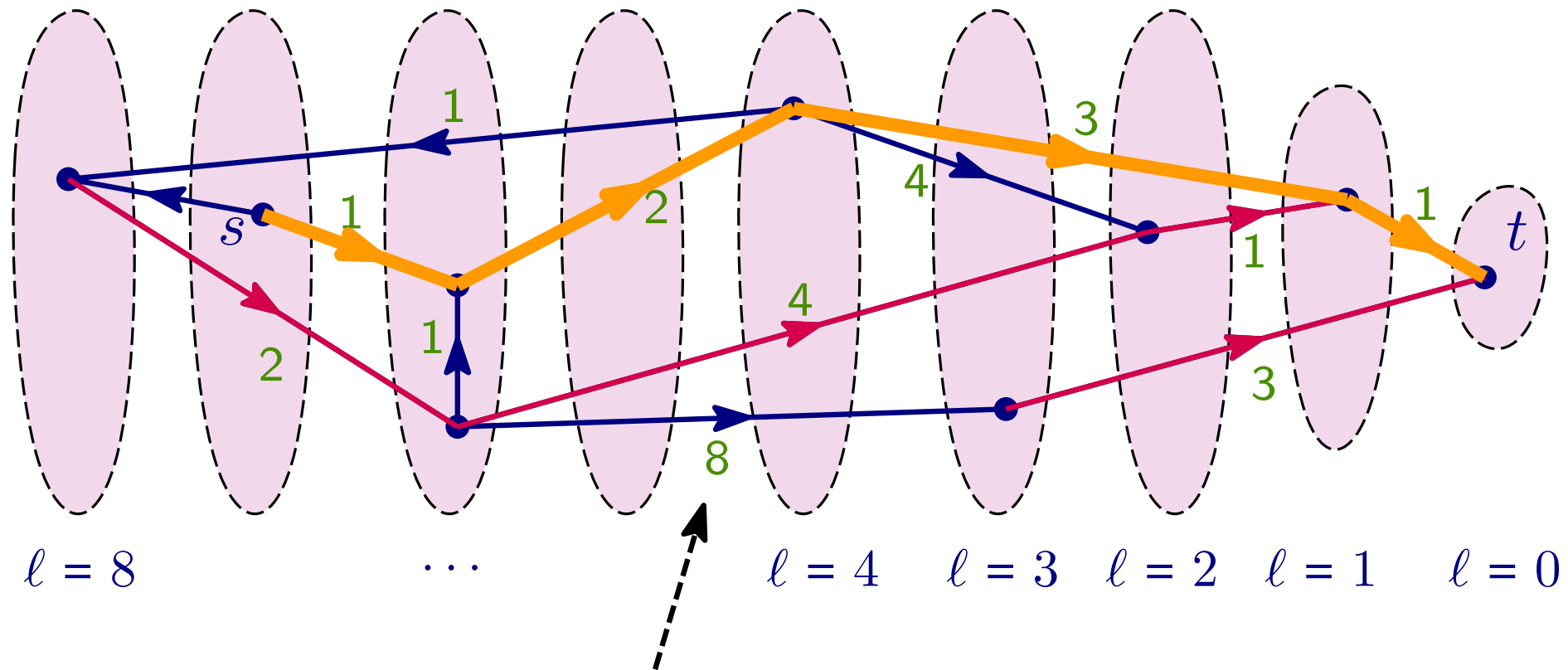


Not all forward edges are admissible!

Weighted Push-Relabel

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

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



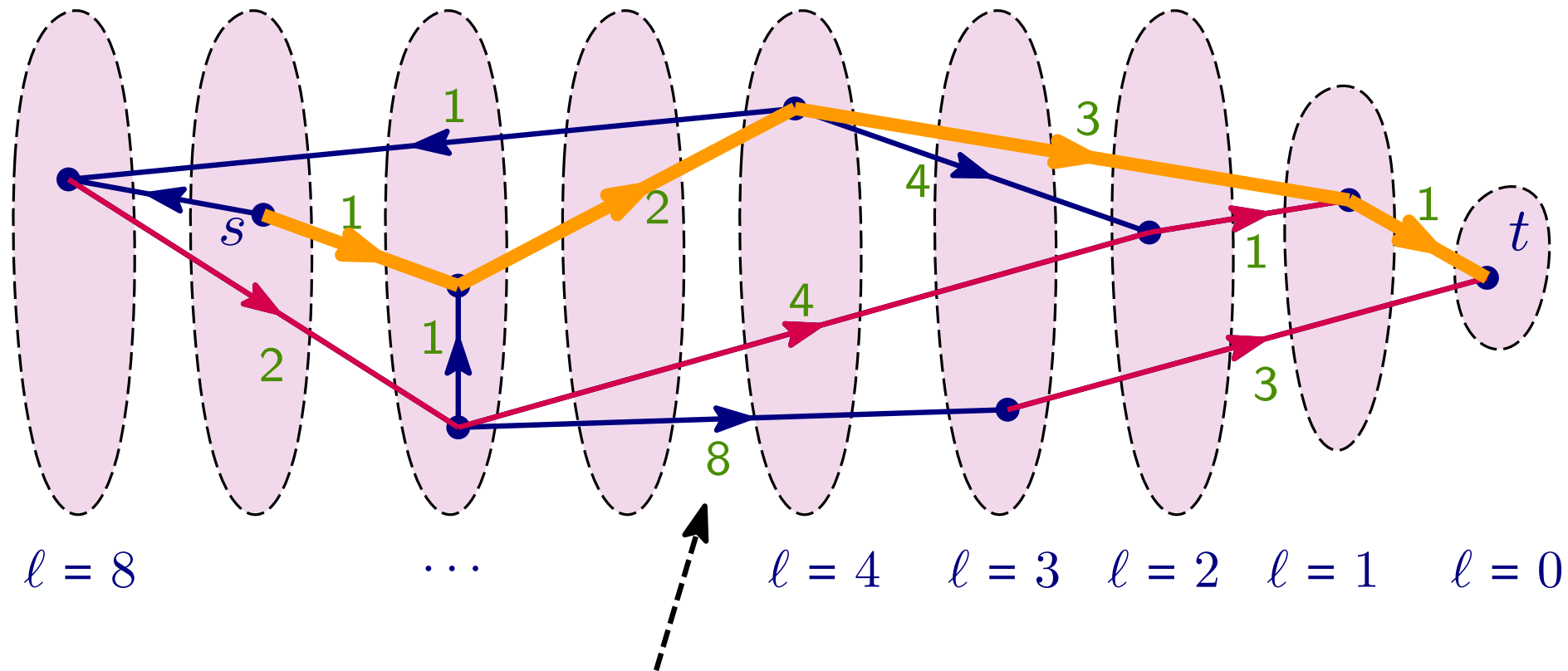
Not all forward edges are admissible!

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

Weighted Push-Relabel

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

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



Not all forward edges are admissible!

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

($\text{\#layers} = \tilde{O}(n)$)

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

After relabel v : recheck only incident edges e where $w(e)$ 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

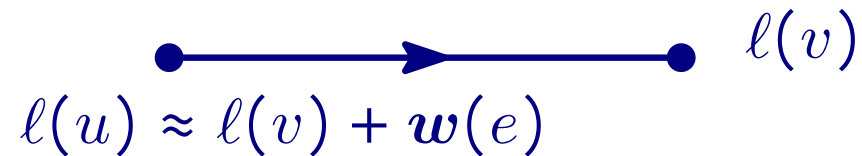
~~$O(nm)$~~ (~~n~~ per edge)

~~$O(nm \log n)$~~ (capacitated graphs:
Link-Cut trees of admissible edges)

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

After relabel v : recheck only incident edges e where $w(e)$ 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

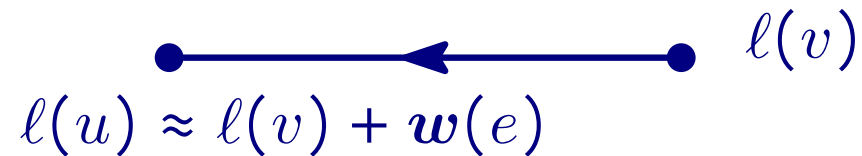
~~$O(nm)$~~ (~~n~~ per edge)

~~$O(nm \log n)$~~ (capacitated graphs:
Link-Cut trees of admissible edges)

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

After relabel v : recheck only incident edges e where $w(e)$ 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

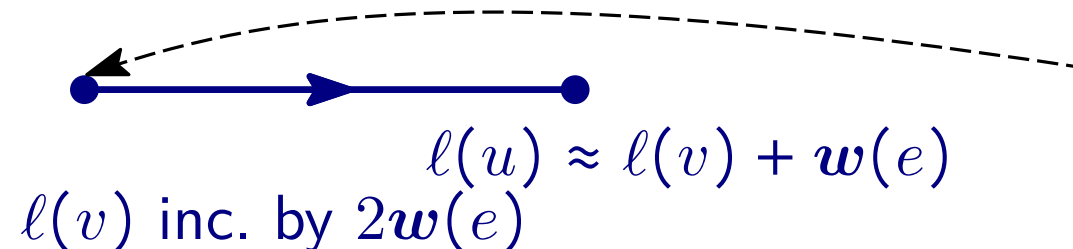
~~$O(nm)$~~ (~~n~~ per edge)

~~$O(nm \log n)$~~ (capacitated graphs:
Link-Cut trees of admissible edges)

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

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

Augmentations



Pseudo-Code

Algorithm 1: PUSHRELABEL($G, c, \Delta, \nabla, w, h$)

```
1 Initialize  $f$  as the empty flow.
2 Let  $\ell(v) = 0$  for all  $v \in V$ . // levels
3 Mark each edge  $e \in \overrightarrow{E} \cup \overleftarrow{E}$  as inadmissible and all vertices as alive.
4 function RELABEL( $v$ )
5   Set  $\ell(v) \leftarrow \ell(v) + 1$ .
6   if  $\ell(v) > 9h$  then
7     mark  $v$  as dead and return.
8   for each edge  $e \ni v$  where  $w(e)$  divides  $\ell(v)$  do
9     Let  $(x, y) = e$ .
10    if  $\ell(x) - \ell(y) \geq 2w(e)$  and  $c_f(e) > 0$  then mark  $e$  as admissible.
11    else mark  $e$  as inadmissible.

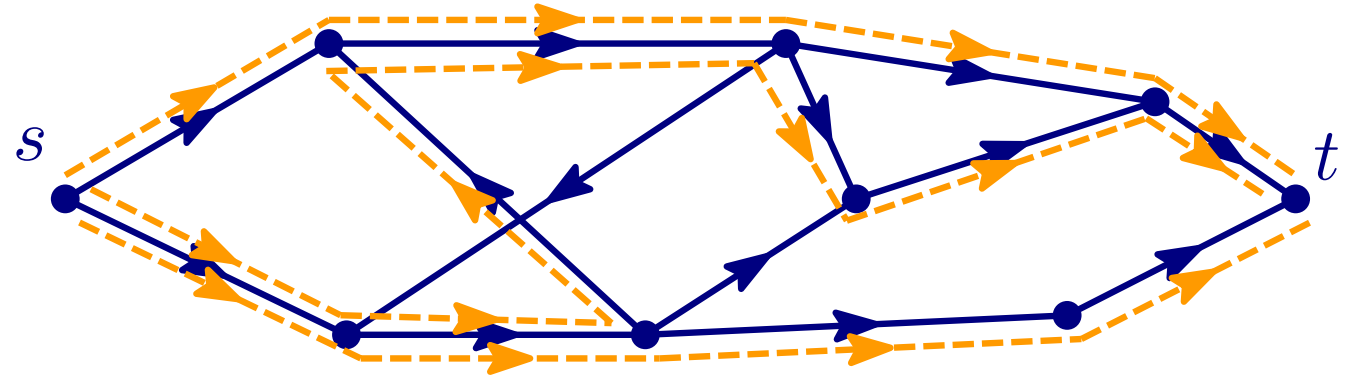
12 main loop
13   while there is an alive vertex  $v$  with  $\nabla_f(v) = 0$  and without an admissible out-edge do
14     RELABEL( $v$ )
15   if there is some alive vertex  $s$  with  $\Delta_f(s) > 0$  then
16     //  $P$  is an "augmenting path"
17     Trace a path  $P$  from  $s$  to some sink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

Similar to normal
Augment-Relabel / Push-Relabel

Good Edge Lengths

- *Good w :*

- $\sum_{e \in E} \frac{n}{w(e)}$ is small ($\approx \tilde{O}(n^2)$, running-time)
- “Optimal” flow f^\star which is short w.r.t. w (flow paths of length $\approx \tilde{O}(n)$)



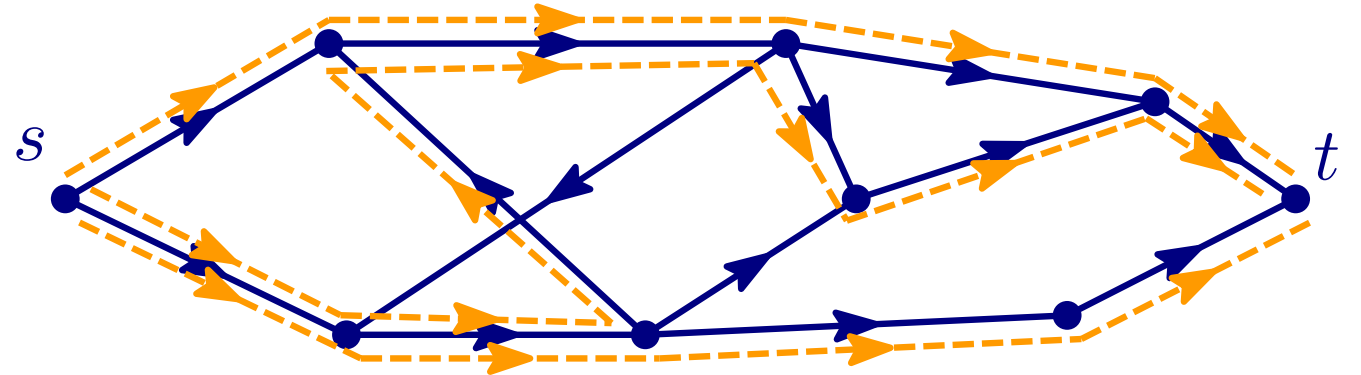
Good Edge Lengths

■ *Good w :*

- $\sum_{e \in E} \frac{n}{w(e)}$ is small ($\approx \tilde{O}(n^2)$, running-time)
- “Optimal” flow f^\star which is short w.r.t. w (flow paths of length $\approx \tilde{O}(n)$)

Lemma.

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



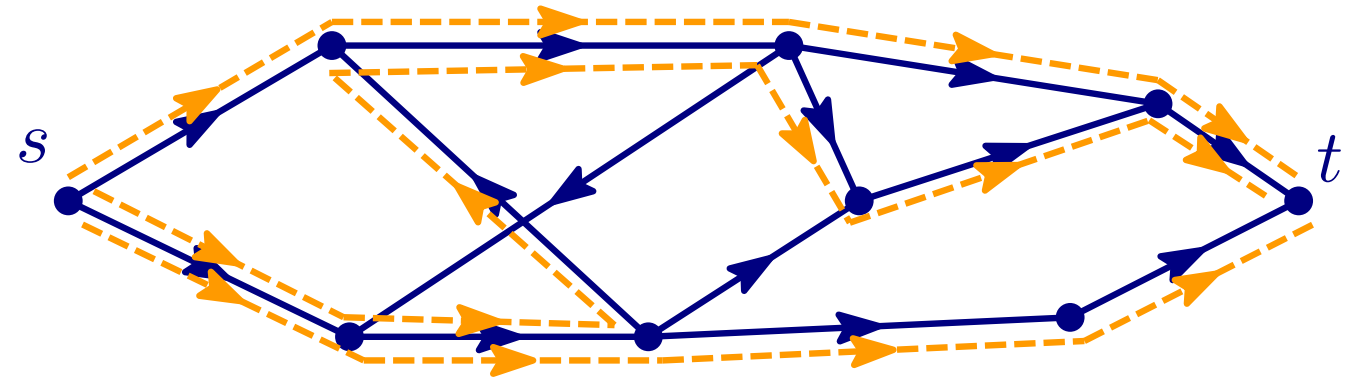
Good Edge Lengths

■ *Good w :*

- $\sum_{e \in E} \frac{n}{w(e)}$ is small ($\approx \tilde{O}(n^2)$, running-time)
- “Optimal” flow f^* which is short w.r.t. w (flow paths of length $\approx \tilde{O}(n)$)

Lemma.

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



Proof Sketch.

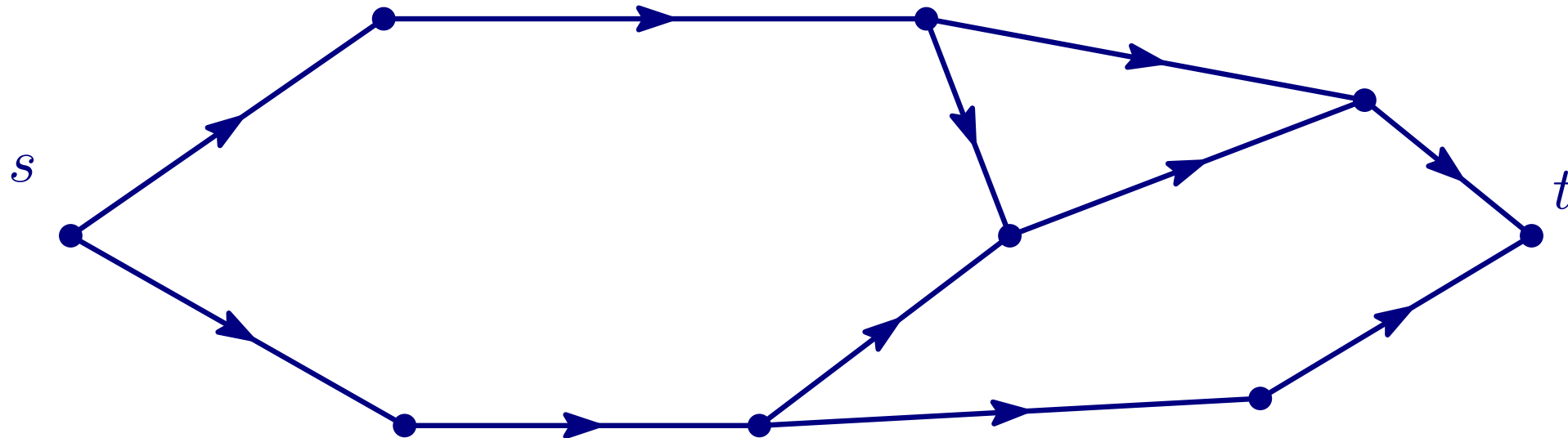
If not: $|f| < \frac{1}{10} |f^*|$

\implies some flow path is still short in residual graph G_f

How to find good edge lengths?

Directed Acyclic Graphs (DAG)

Def: no directed cycles

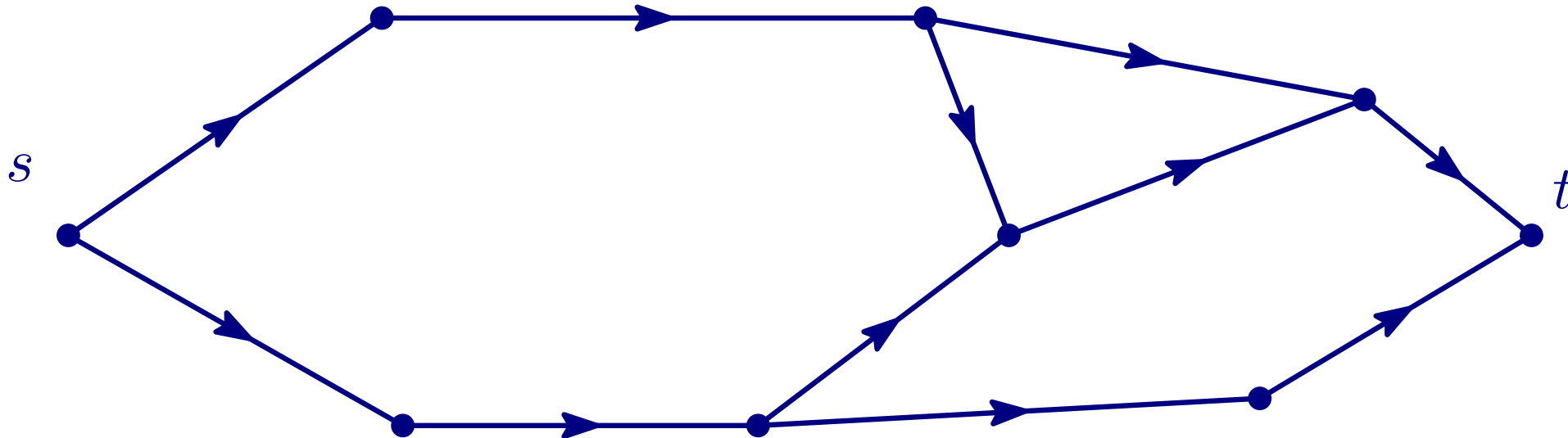


Directed Acyclic Graphs (DAG)

Def: no directed cycles

Good edge lengths w ?

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



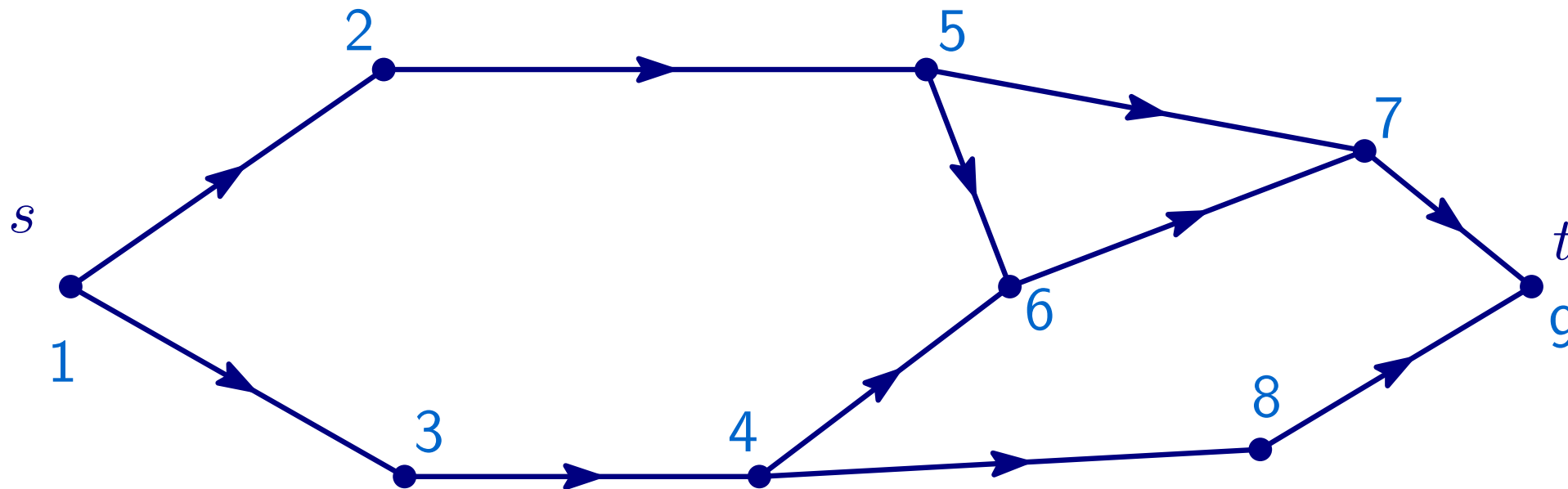
Directed Acyclic Graphs (DAG)

Def: no directed cycles

Topological order τ

Good edge lengths w ?

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



Directed Acyclic Graphs (DAG)

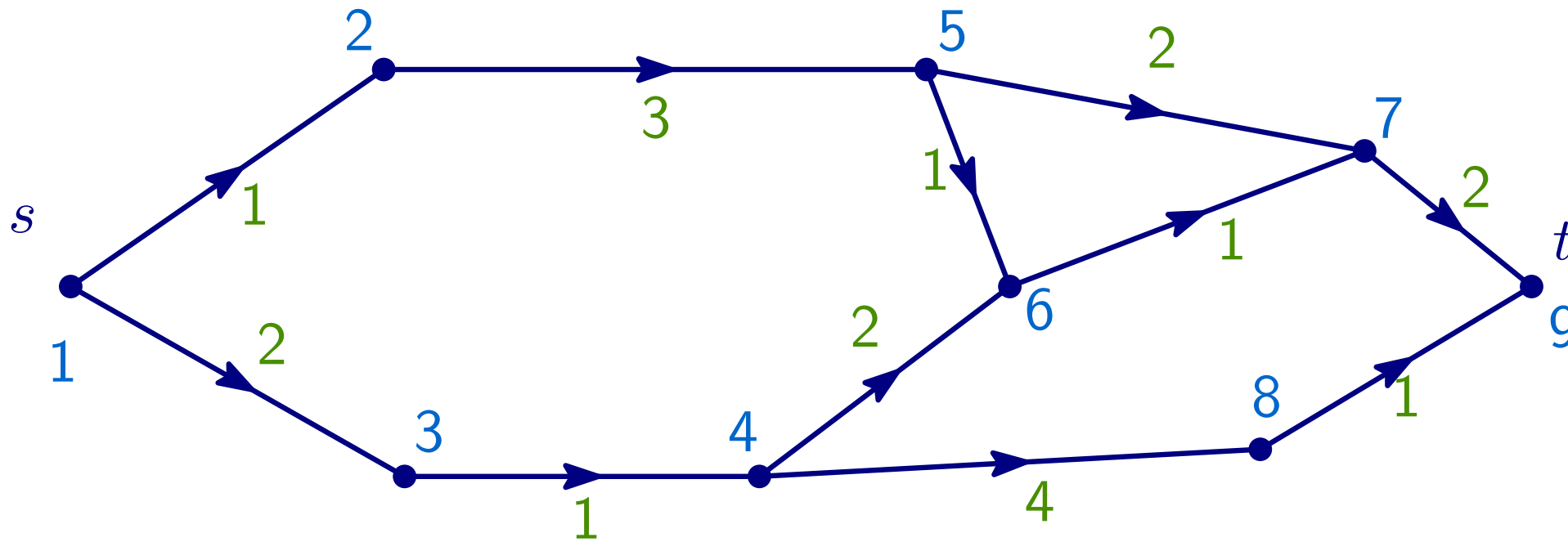
Def: no directed cycles

Topological order τ

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

Good edge lengths w ?

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



Directed Acyclic Graphs (DAG)

Def: no directed cycles

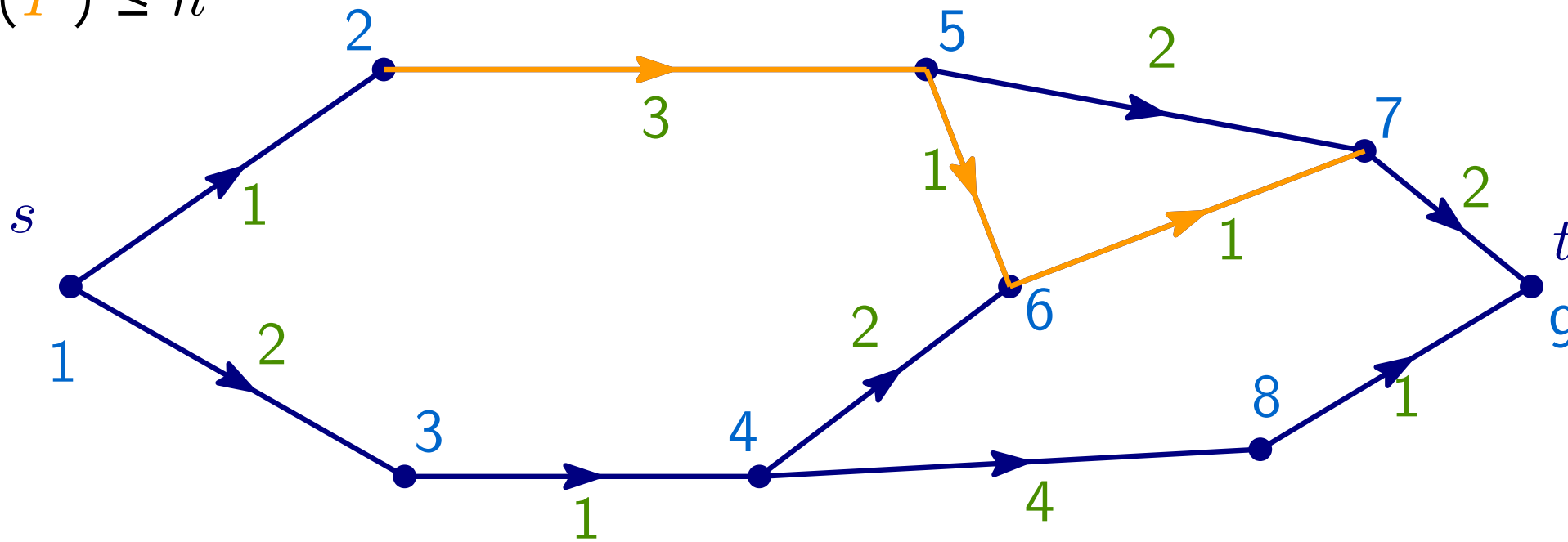
Topological order τ

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

$$w(P) \leq n$$

Good edge lengths w ?

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



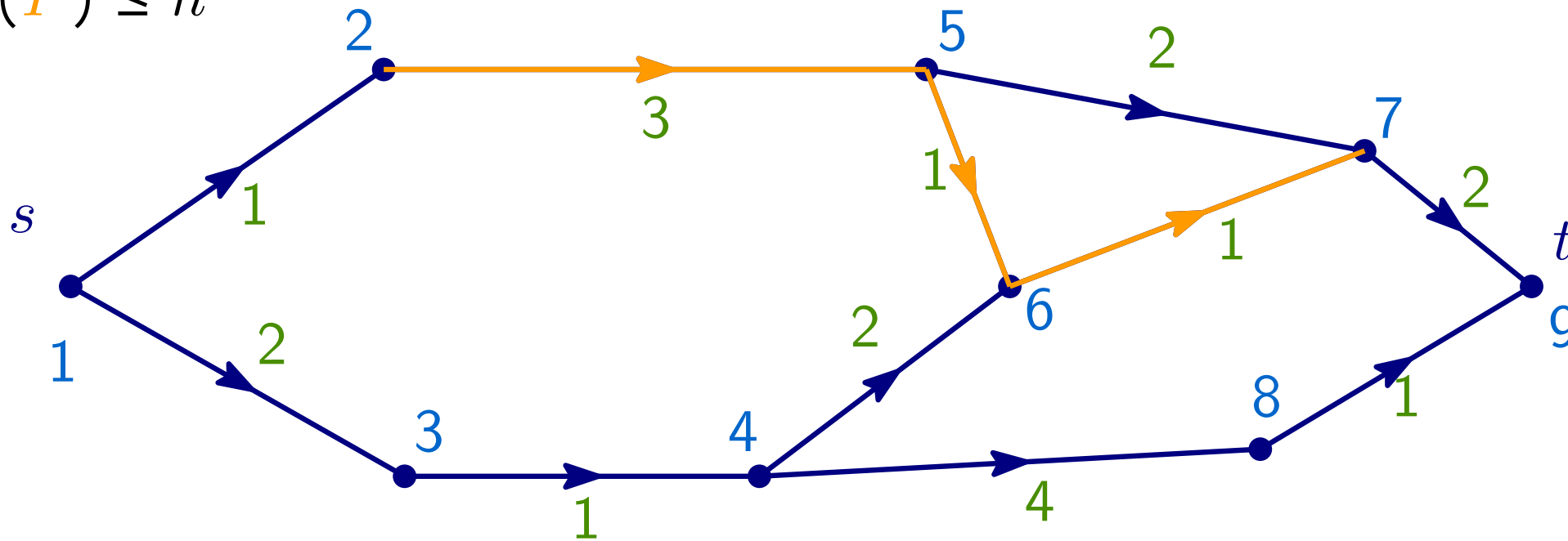
Directed Acyclic Graphs (DAG)

Def: no directed cycles

Topological order τ

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

$$w(P) \leq n$$



Good edge lengths w ?



■ $\sum_{e \in E} \frac{n}{w(e)}$ is small



■ “Optimal” flow f^* which is short w.r.t. w

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

Directed Acyclic Graphs (DAG)

Def: no directed cycles

Topological order τ

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

$$w(P) \leq n$$

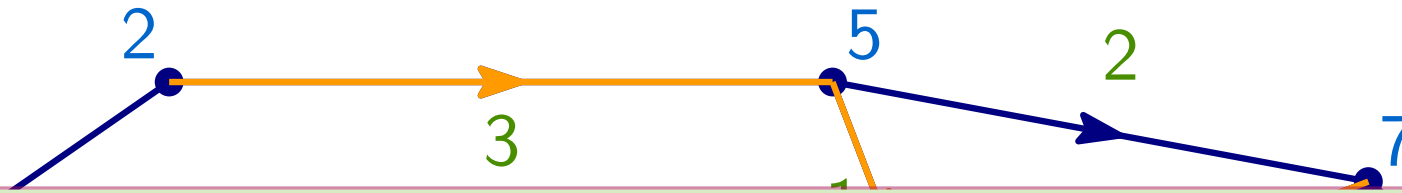
Good edge lengths w ?



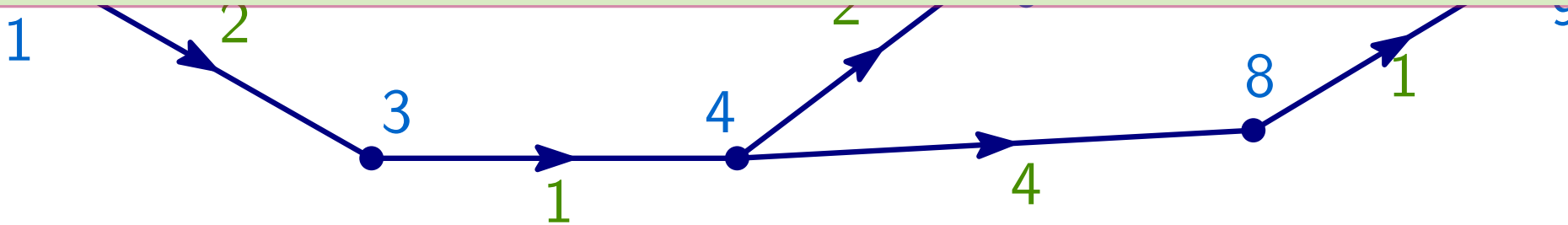
- $\sum_{e \in E} \frac{n}{w(e)}$ is small



- “Optimal” flow f^* which is short w.r.t. w



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



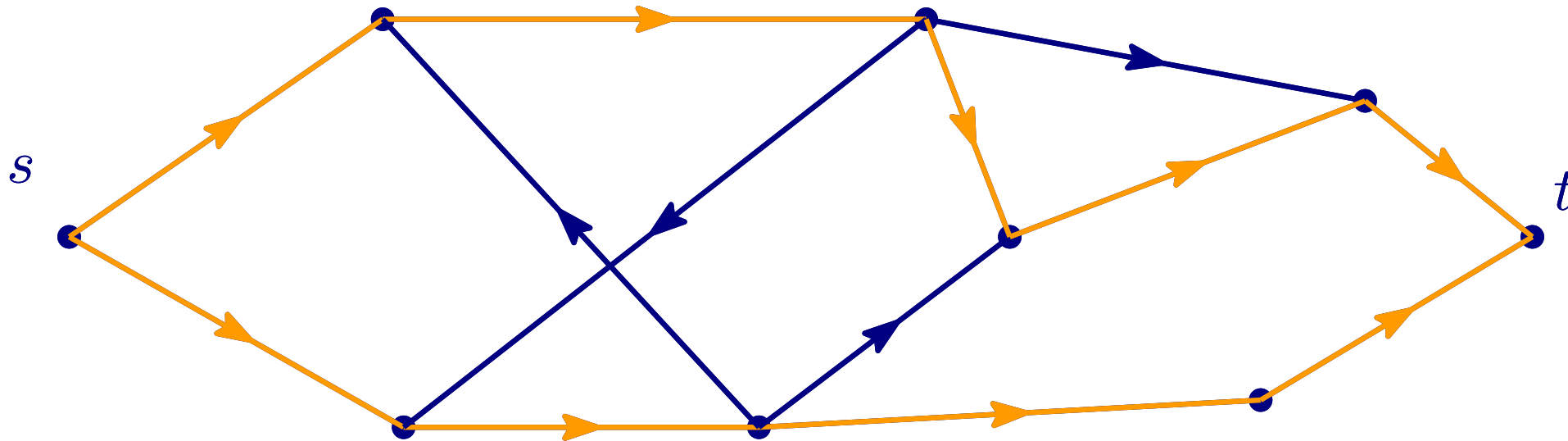
$$\sum_{(u,v)} \frac{n}{|\tau(u) - \tau(v)|} \leq n \sum_{v \in V} \sum_{k=1}^{n-1} \frac{1}{k} \leq n^2 \log n$$

General Graphs — Attempt One

1. Compute maxflow f^\star

Good edge lengths w ?

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

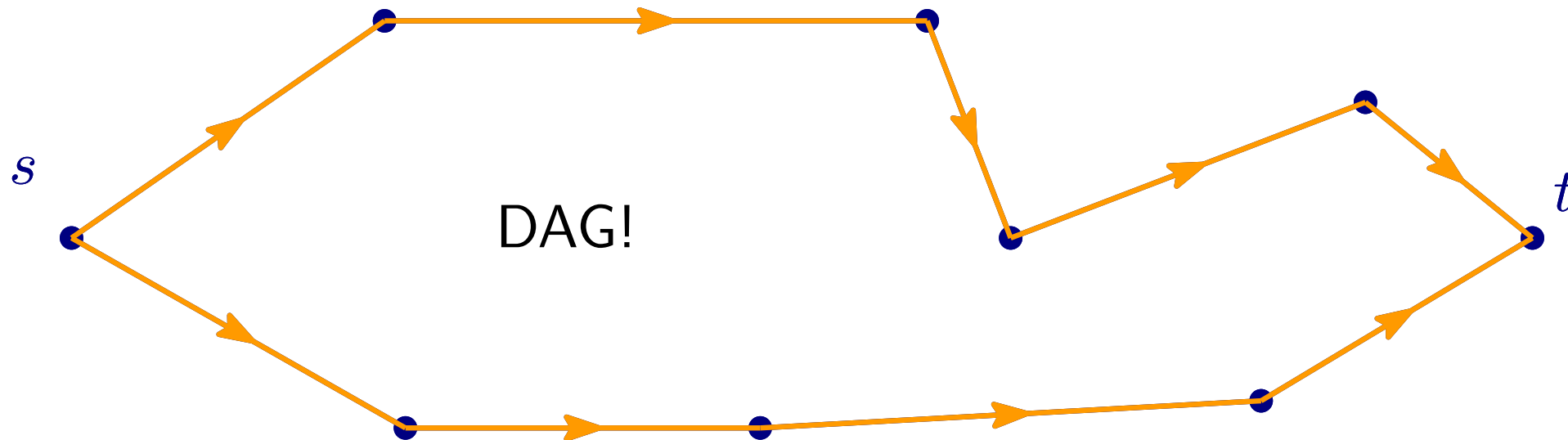


General Graphs — Attempt One

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

Good edge lengths w ?

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

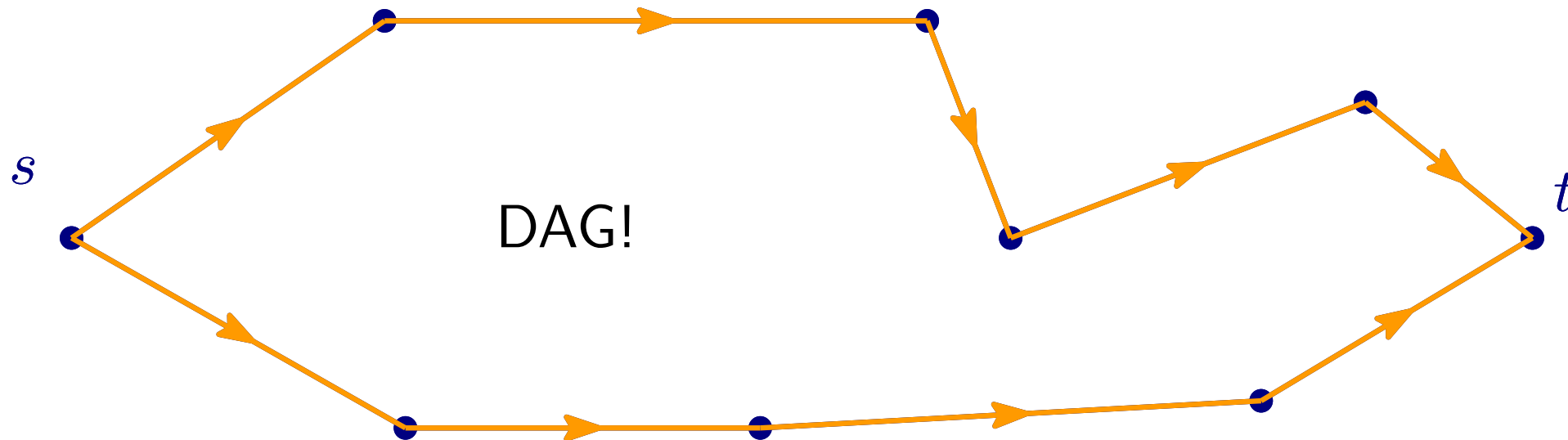


General Graphs — Attempt One

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

Good edge lengths w ?

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

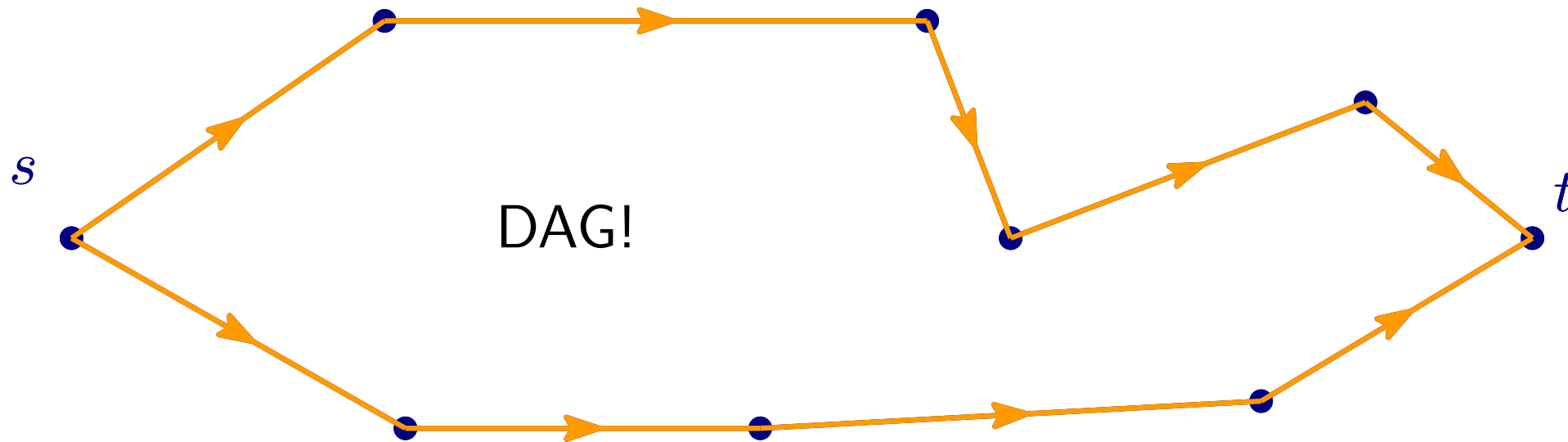


General Graphs — Attempt One

1. Compute maxflow f^\star
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 lengths w ?

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

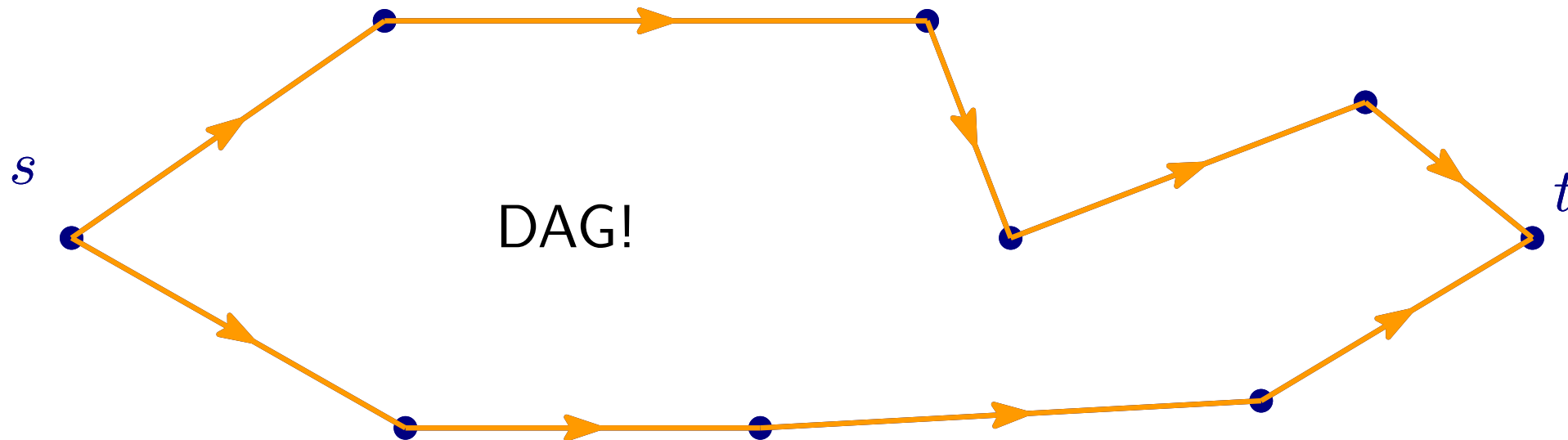


General Graphs — Attempt One

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

Good edge lengths w ?

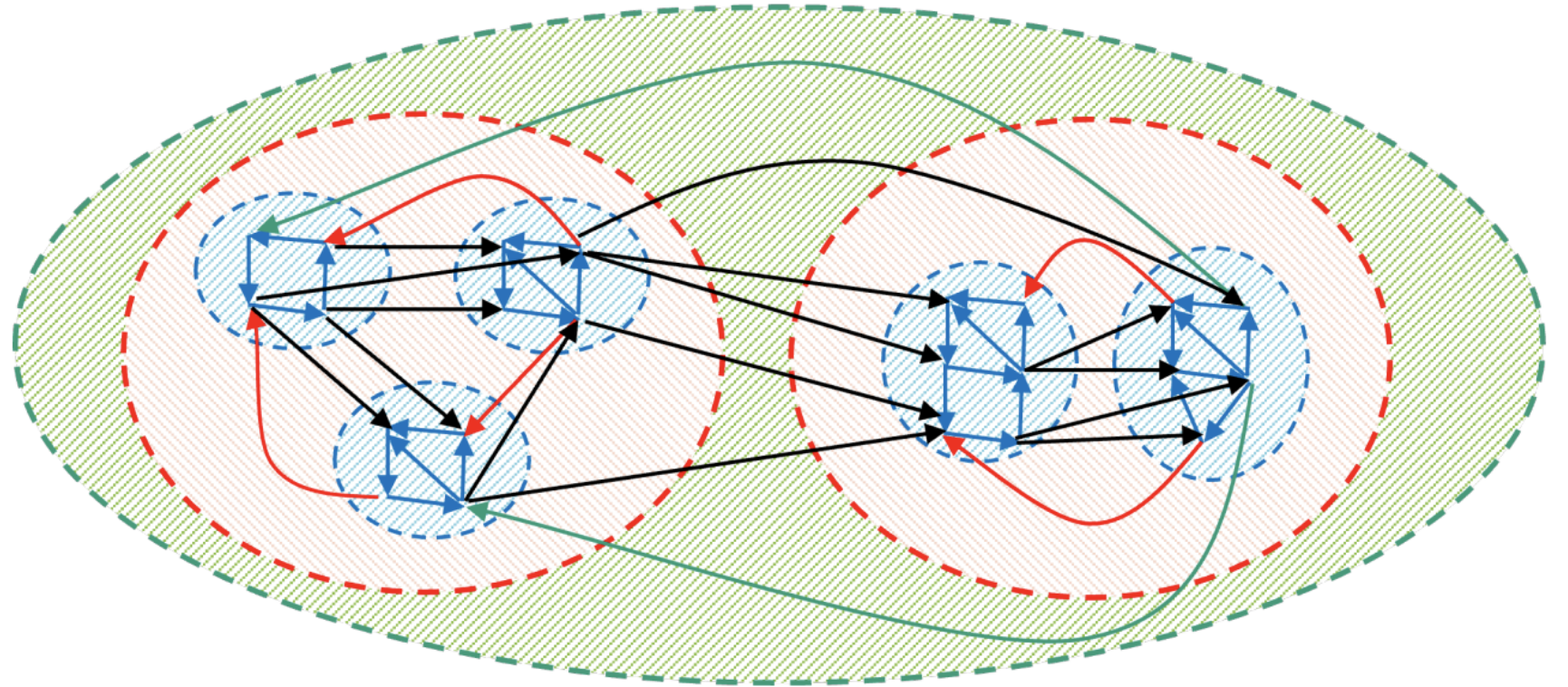
- $\sum_{e \in E} \frac{n}{w(e)}$ is small
- “Optimal” flow f^* which is short w.r.t. w



General Graphs

Good edge lengths w ?

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



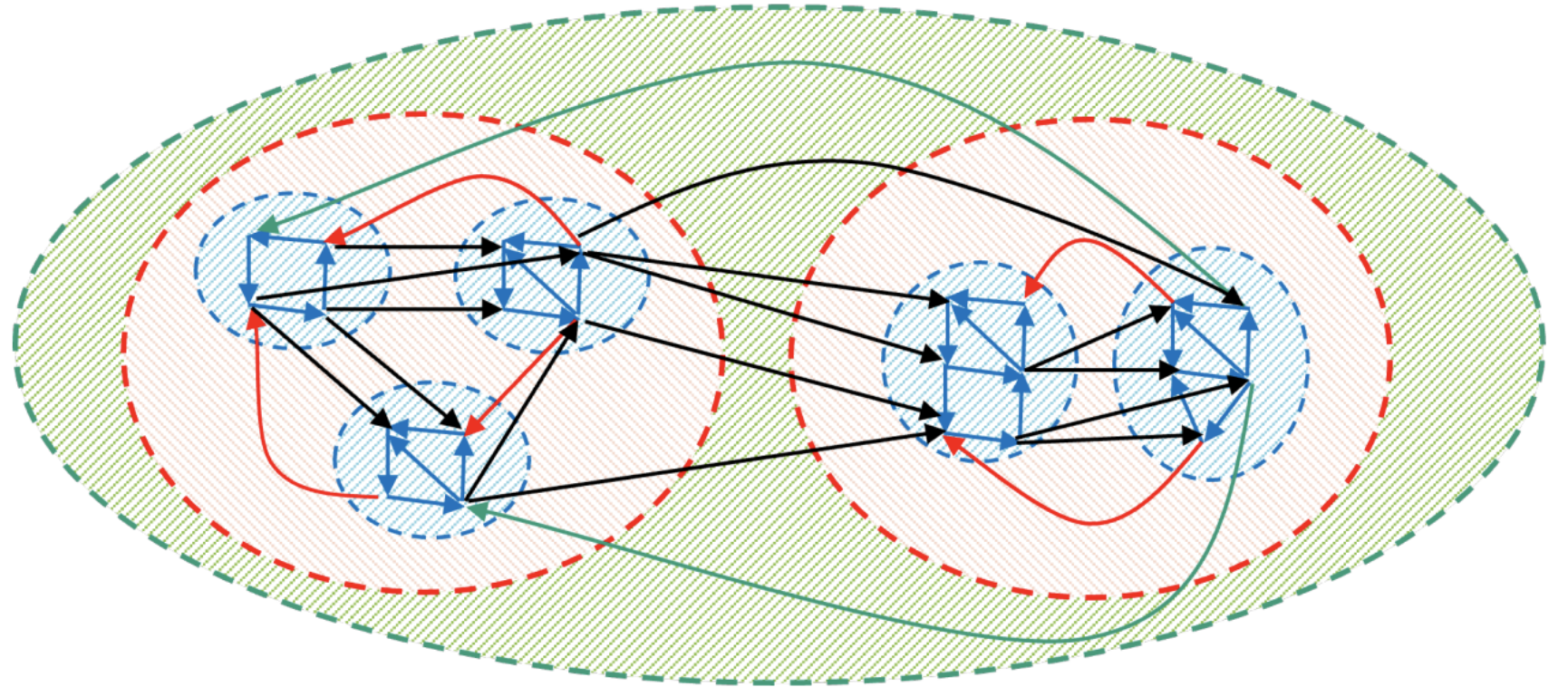
General Graphs

“Pseudo-Topological” order τ

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

Good edge lengths w ?

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



General Graphs

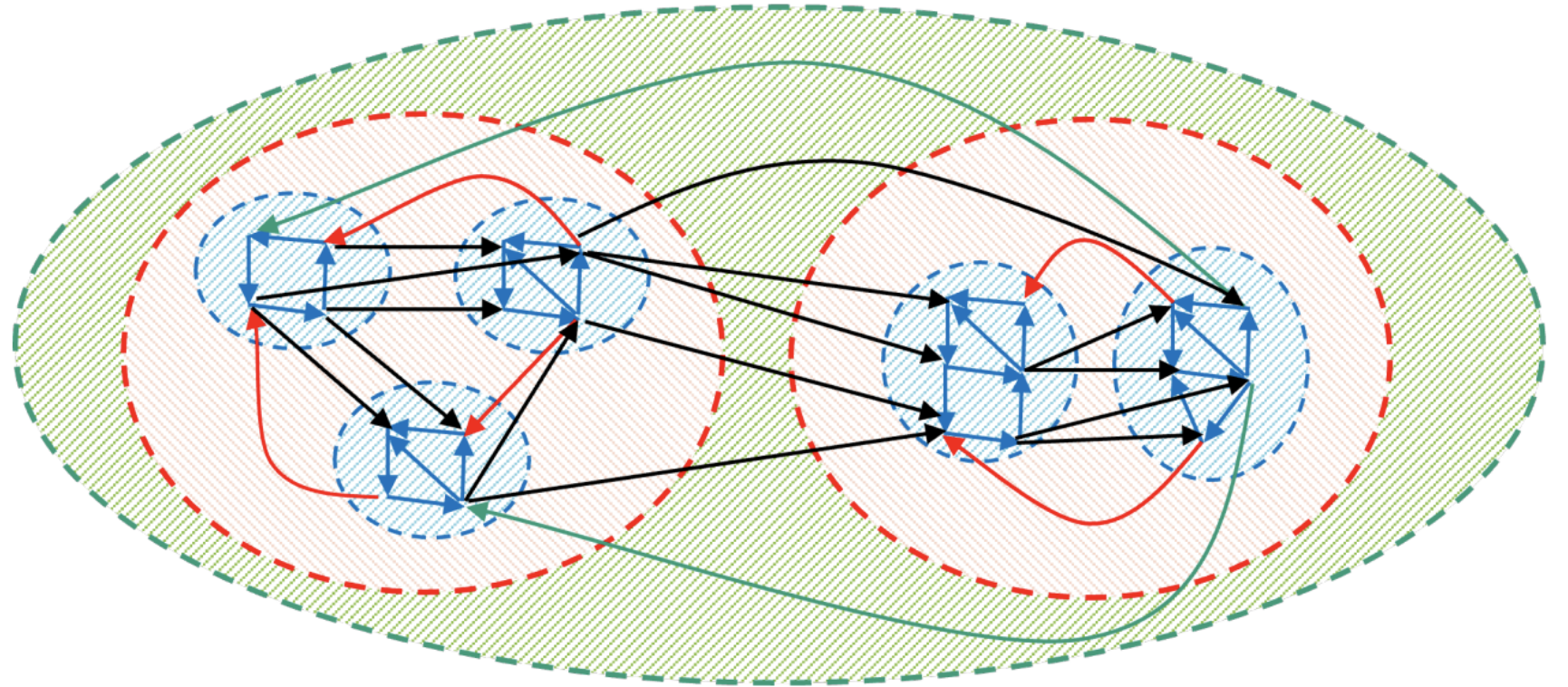
“Pseudo-Topological” order τ

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

Directed Expander Hierarchy

Good edge lengths w ?

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



General Graphs

“Pseudo-Topological” order τ

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

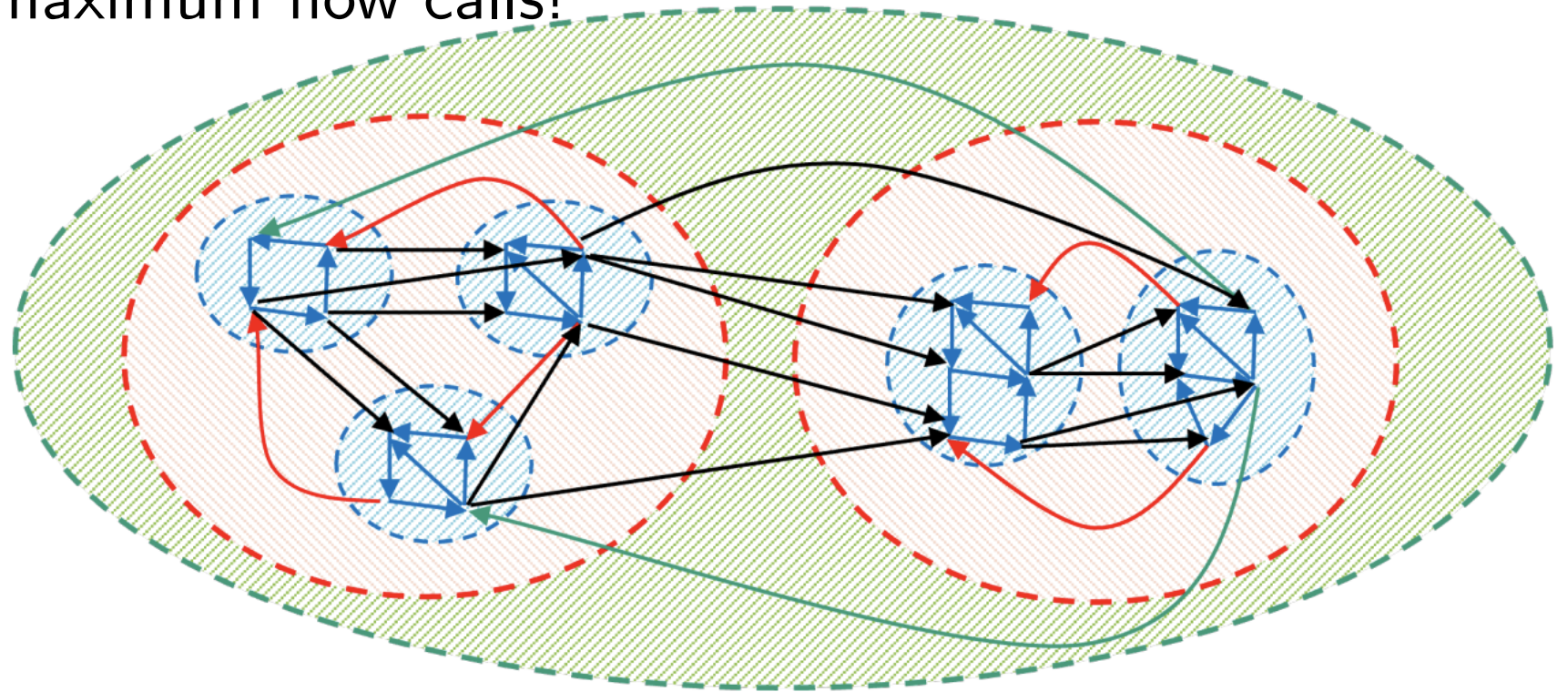
Directed Expander Hierarchy

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

(Cheating!)

Good edge lengths w ?

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



General Graphs

“Pseudo-Topological” order τ

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

Directed Expander Hierarchy

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

(Cheating!)

Instead:

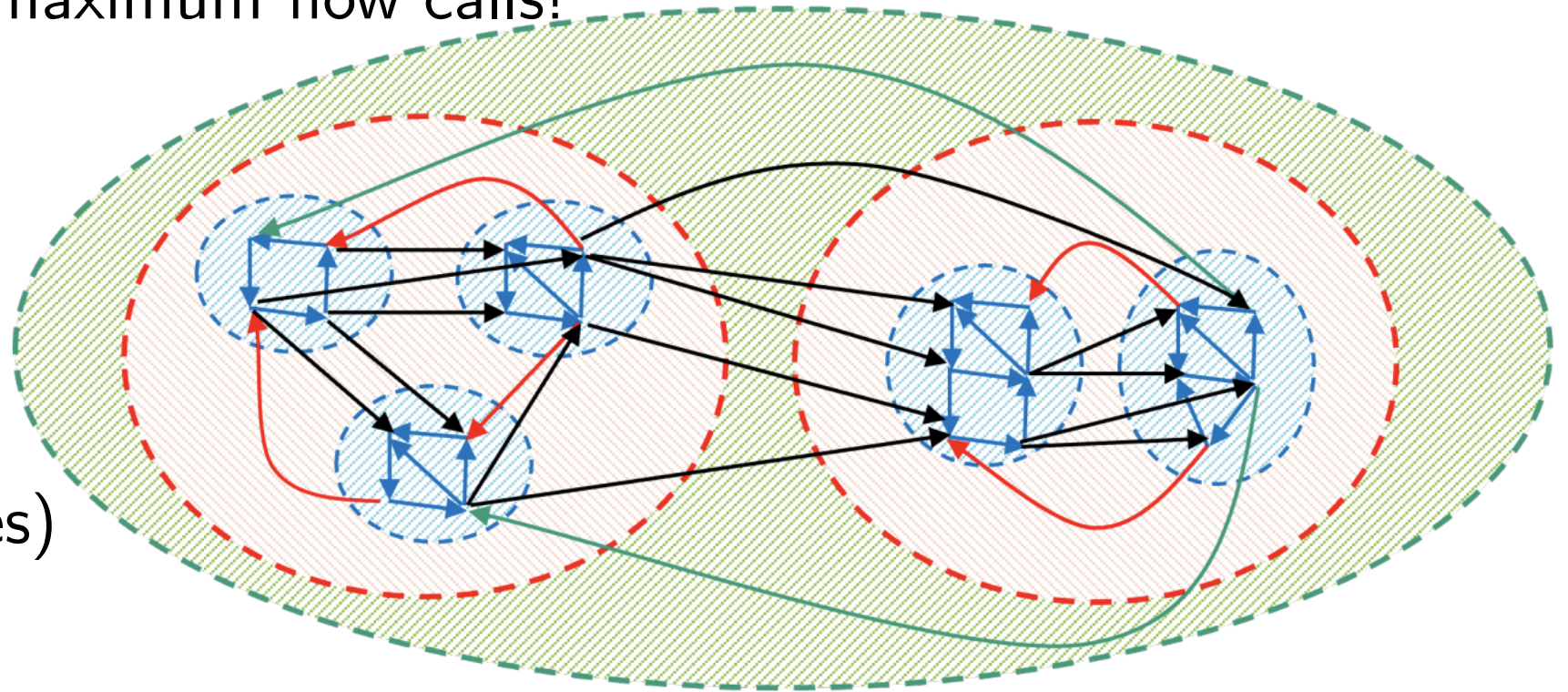
Build Bottom Up

Bootstrap Weighted P.R.

(solve “easier” flow instances)

Good edge lengths w ?

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



General Graphs

“Pseudo-Topological” order τ

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

Directed Expander Hierarchy

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

(Cheating!)

Instead:

Build Bottom Up

Bootstrap Weighted P.R.

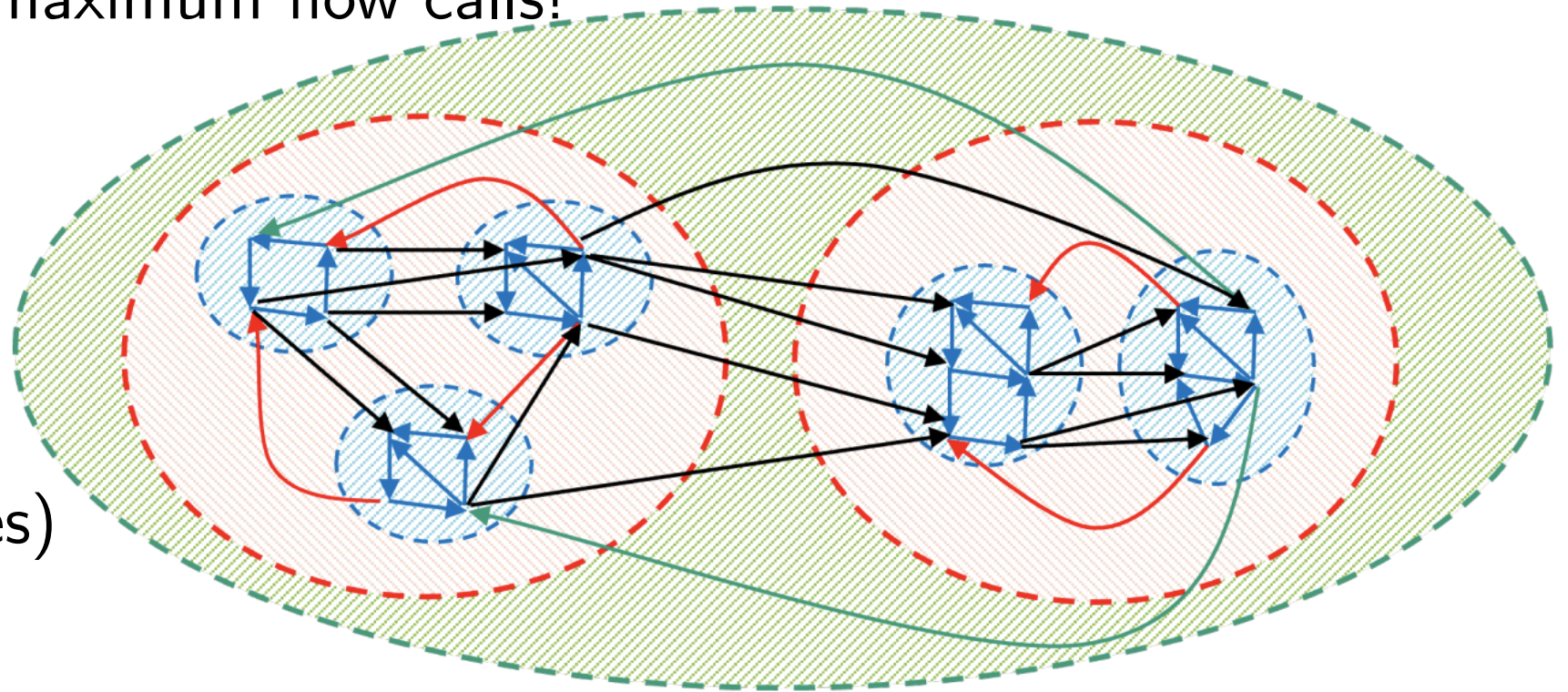
(solve “easier” flow instances)

Technically Complicated :(
(bad guy: nestedness)

(half of the 99 page paper)

Good edge lengths w ?

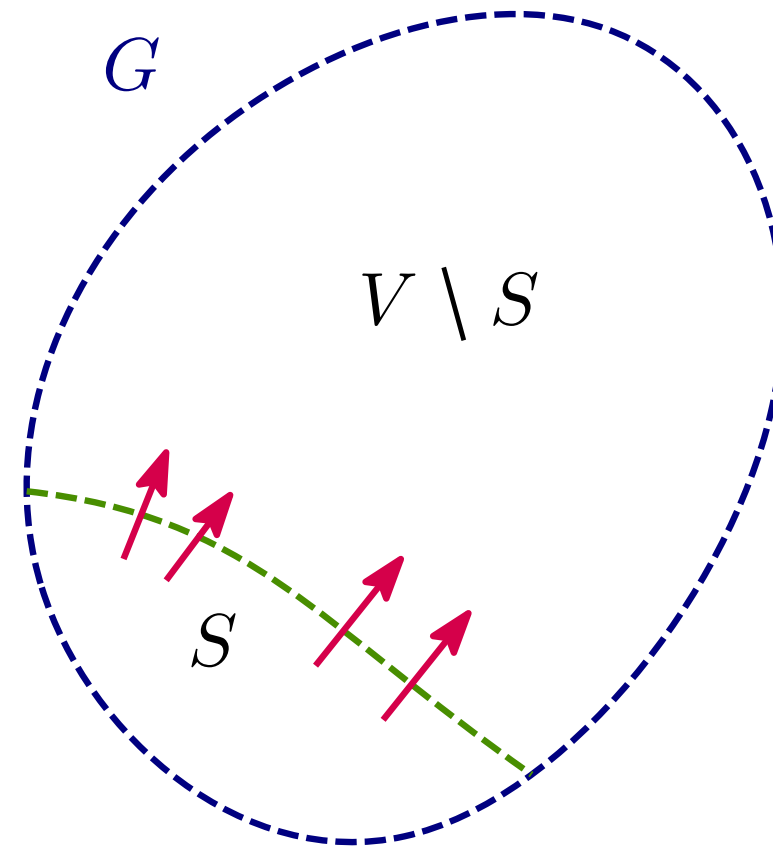
- ✓ $\sum_{e \in E} \frac{n}{w(e)}$ is small
- “Optimal” flow f^* which is short w.r.t. w



(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 $E(S, V \setminus S) \geq \phi \cdot \min\{\text{vol}(S), \text{vol}(V \setminus S)\} \quad \forall S$

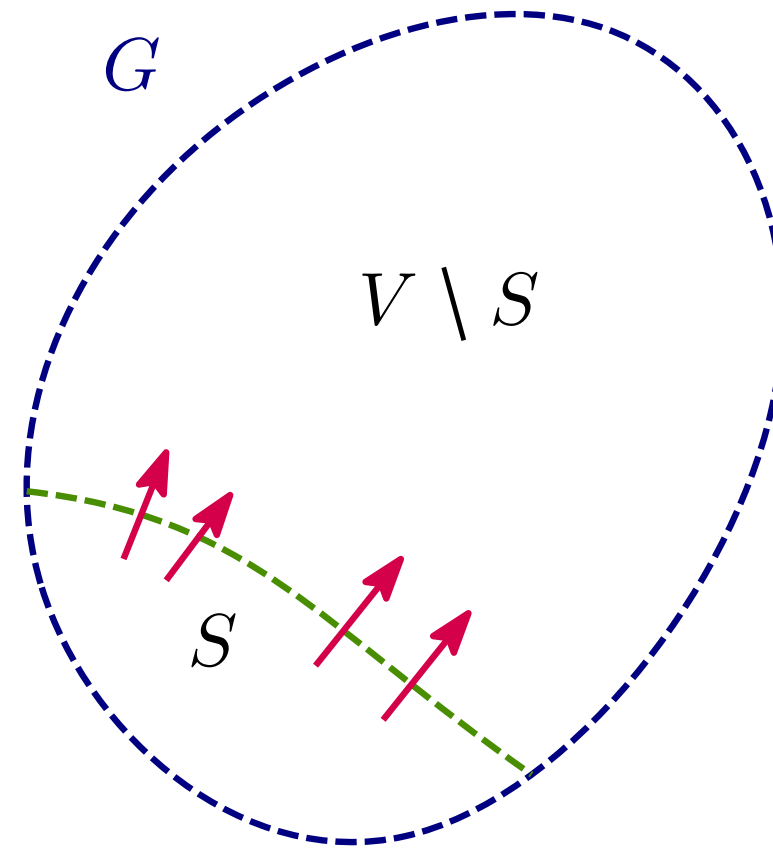
($\text{vol}(S) = \sum_{v \in S} \deg(v)$, $\phi \approx 1/n^{o(1)}$)

Examples:

Cliques

Bidirected Stars

Random



(Directed) Expanders

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

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

Examples:

Cliques

Bidirected Stars

Random

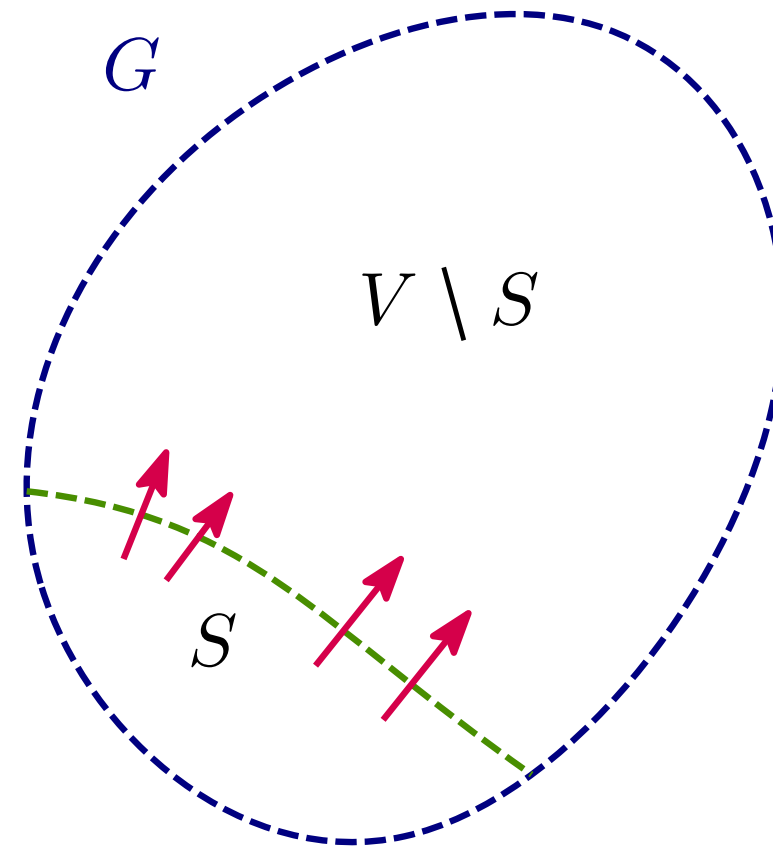
Why?

Well-connected

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

Easy to route (short) flow in

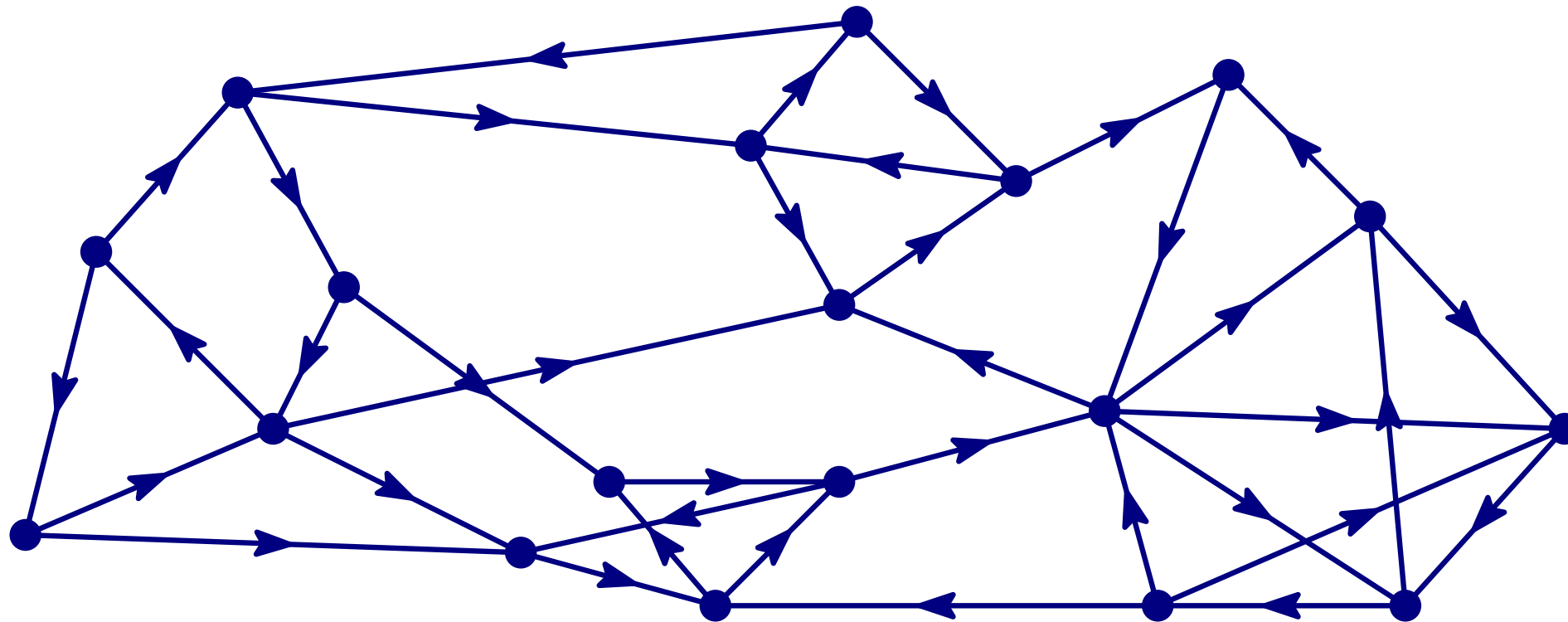
Robust to small changes



(Directed) Expander Decomposition

Every graph can be decomposed into:

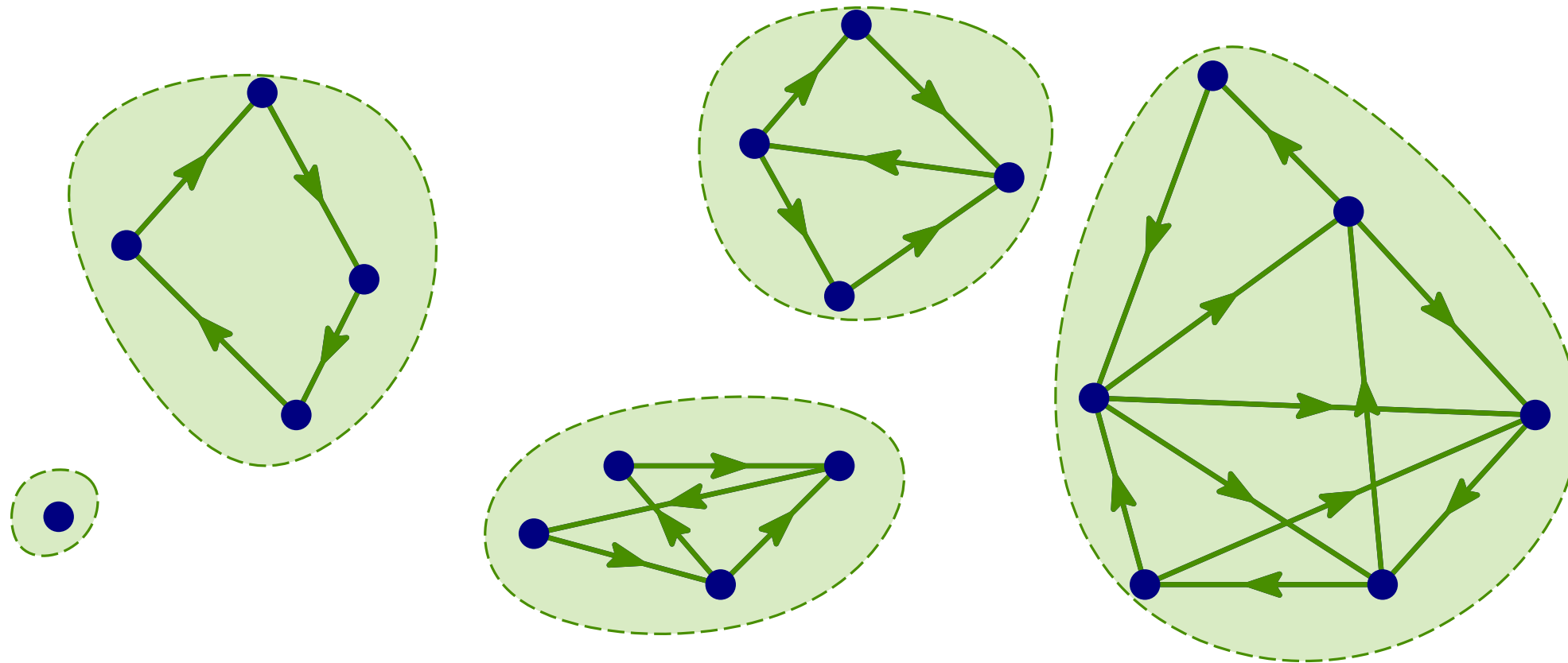
- 1.
- 2.
- 3.



(Directed) Expander Decomposition

Every graph can be decomposed into:

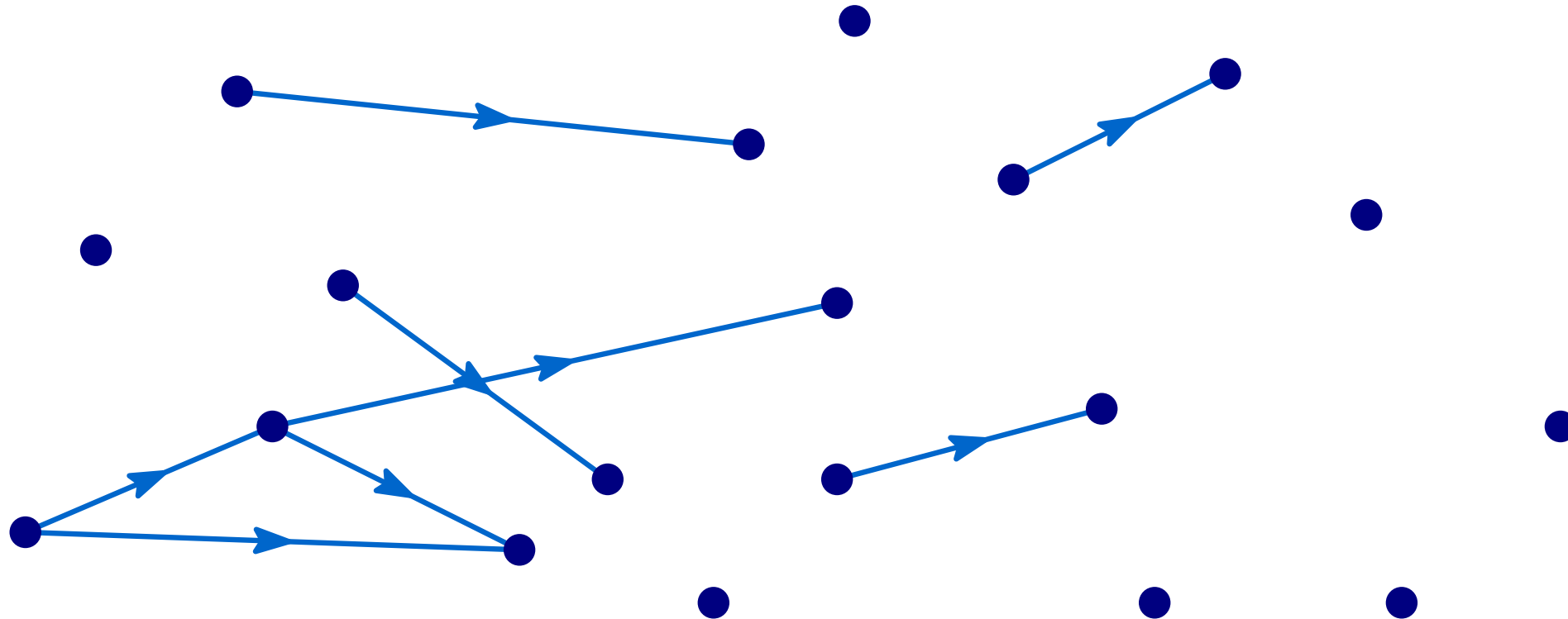
1. Expanders X_1, \dots, X_k
- 2.
- 3.



(Directed) Expander Decomposition

Every graph can be decomposed into:

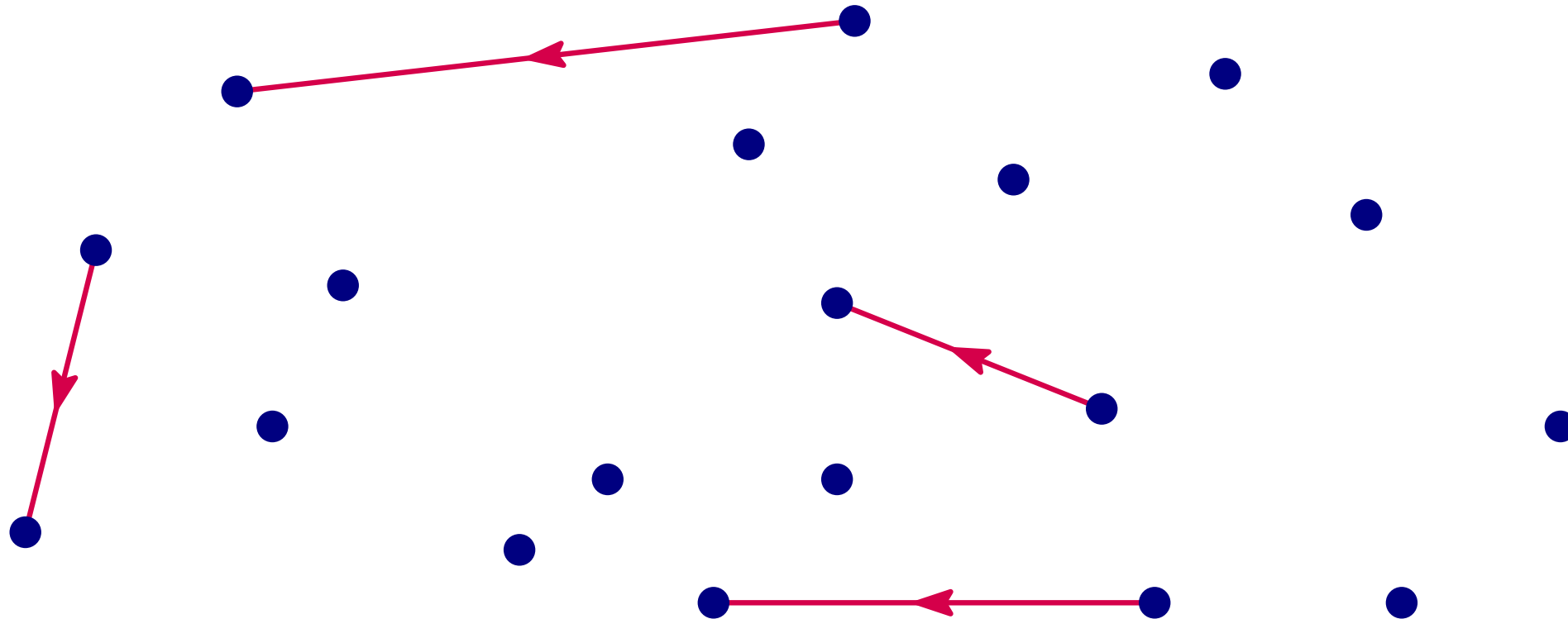
1. Expanders X_1, \dots, X_k
2. DAG edges D
- 3.



(Directed) Expander Decomposition

Every graph can be decomposed into:

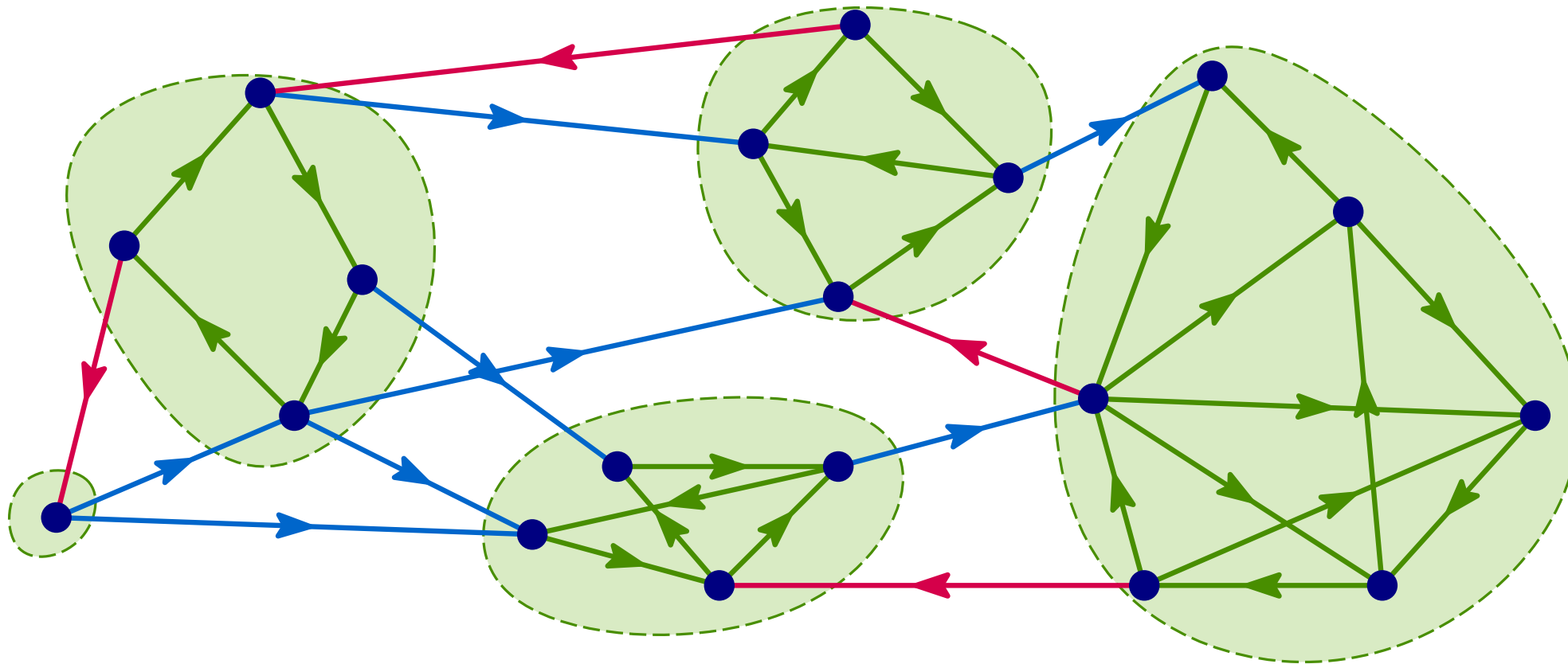
1. Expanders X_1, \dots, X_k
2. DAG edges D
3. Few backward edges B



(Directed) Expander Decomposition

Every graph can be decomposed into:

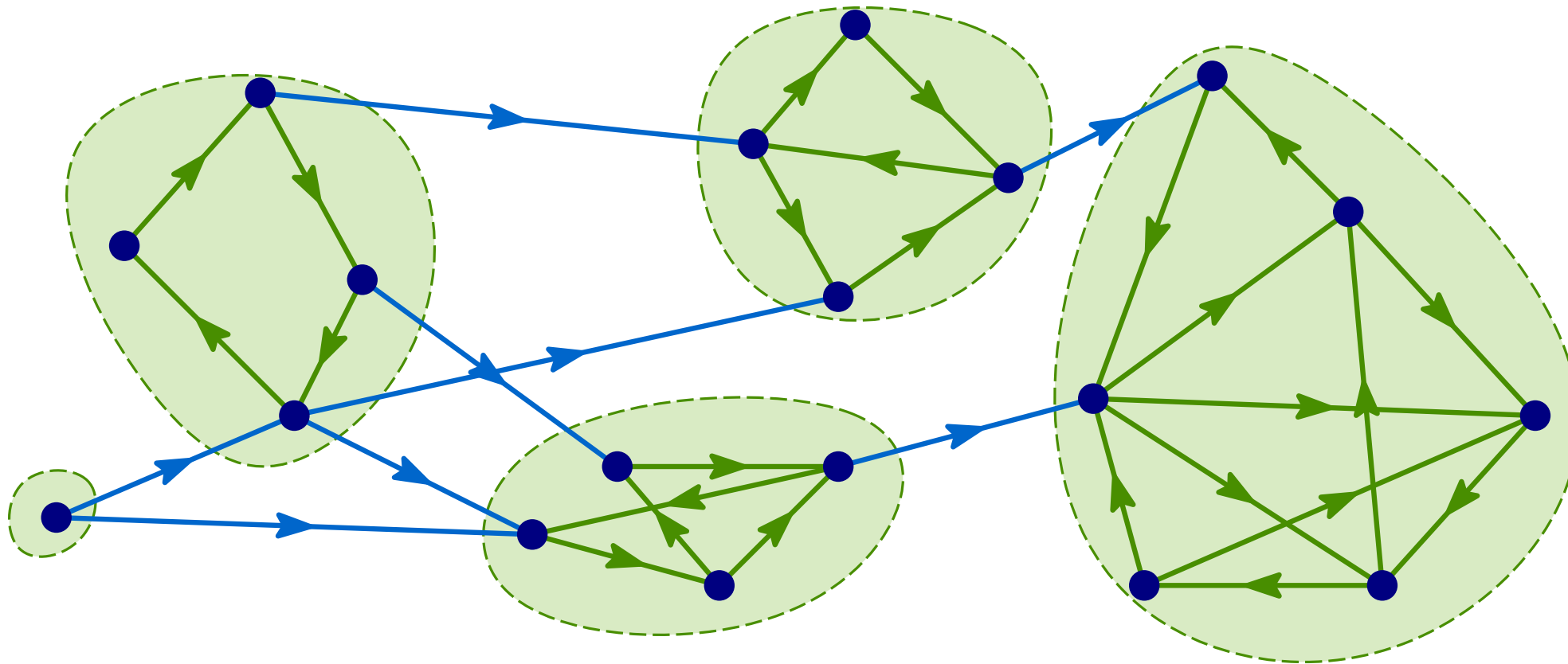
1. Expanders X_1, \dots, X_k
2. DAG edges D
3. Few backward edges B



(Directed) Expander Decomposition

Every graph can be decomposed into:

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



(Directed) Expander Decomposition

Every graph can be decomposed into:

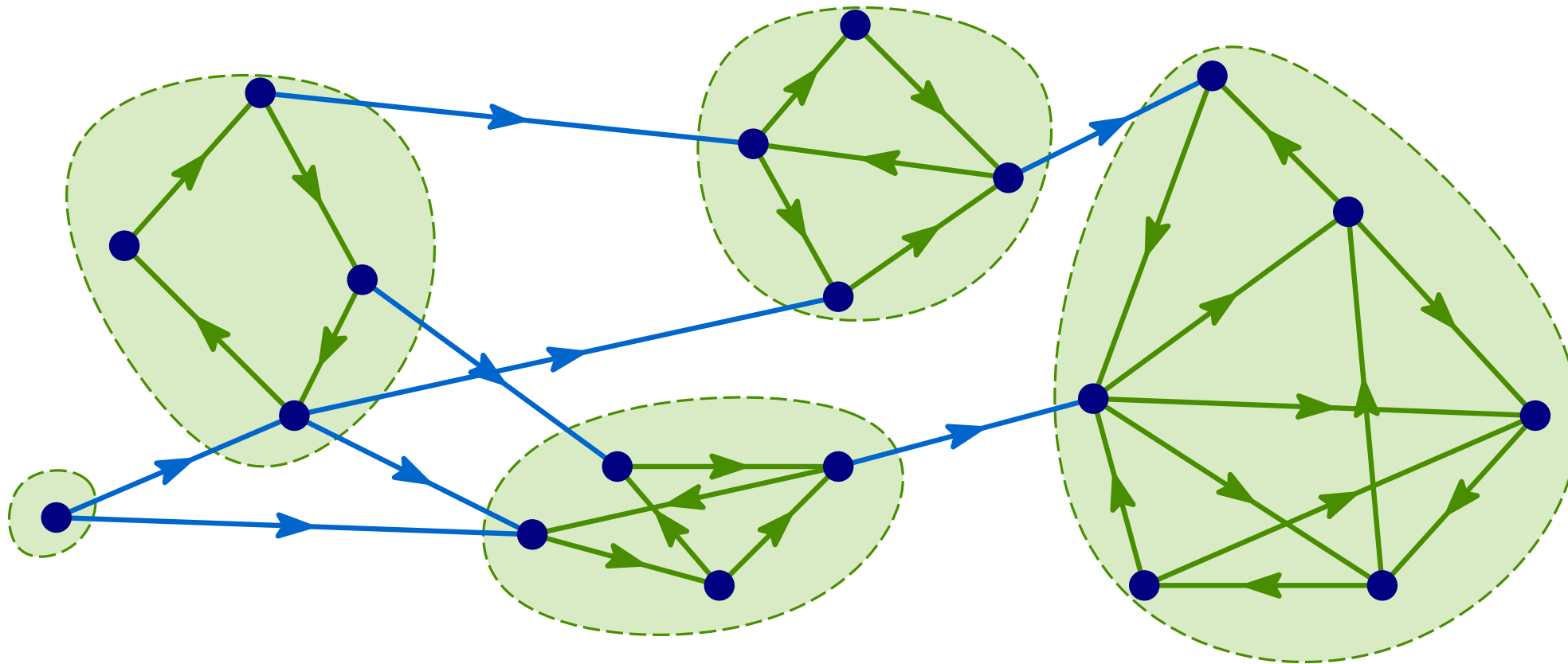
1. Expanders $X_1, \dots, X_k = \text{SCC}(G \setminus B)$
2. DAG edges D
3. Few backward edges B

Good edge lengths in $G \setminus B$:

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

τ respects DAG

τ contiguous in expanders



(Directed) Expander Decomposition

Every graph can be decomposed into:

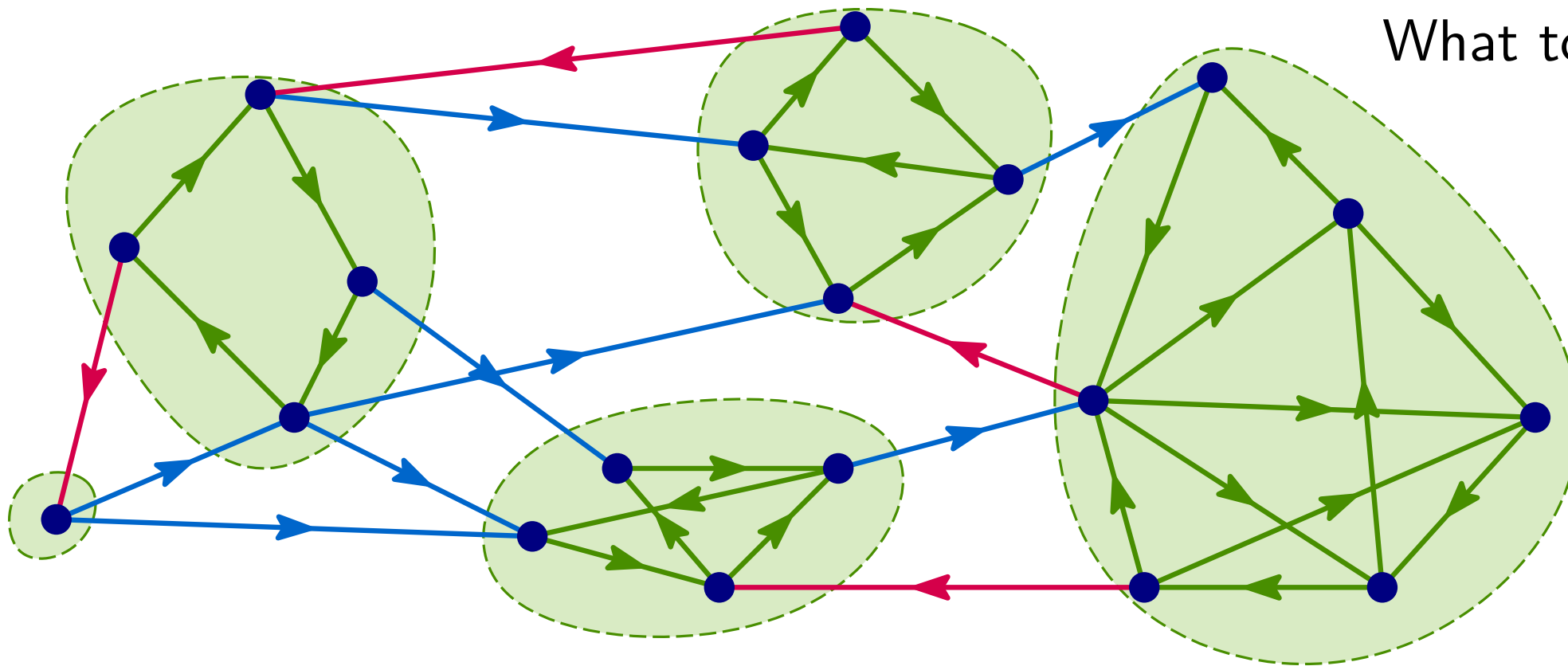
1. Expanders $X_1, \dots, X_k = \text{SCC}(G \setminus B)$
2. DAG edges D
3. Few backward edges B

Good edge lengths in $G \setminus B$:

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

τ respects DAG

τ contiguous in expanders



(Directed) Expander Decomposition

Every graph can be decomposed into:

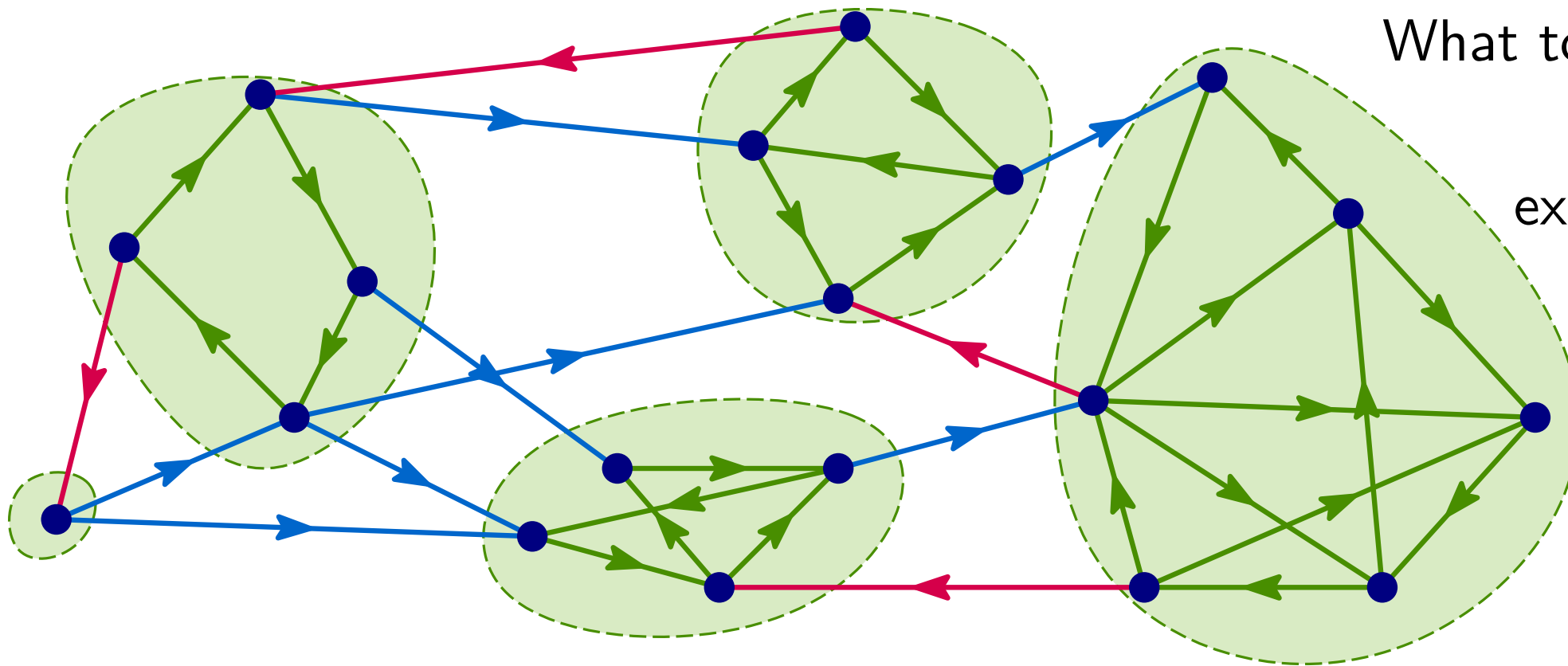
1. Expanders $X_1, \dots, X_k = \text{SCC}(G \setminus B)$
2. DAG edges D
3. Few backward edges B

Good edge lengths in $G \setminus B$:

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

τ respects DAG

τ contiguous in expanders



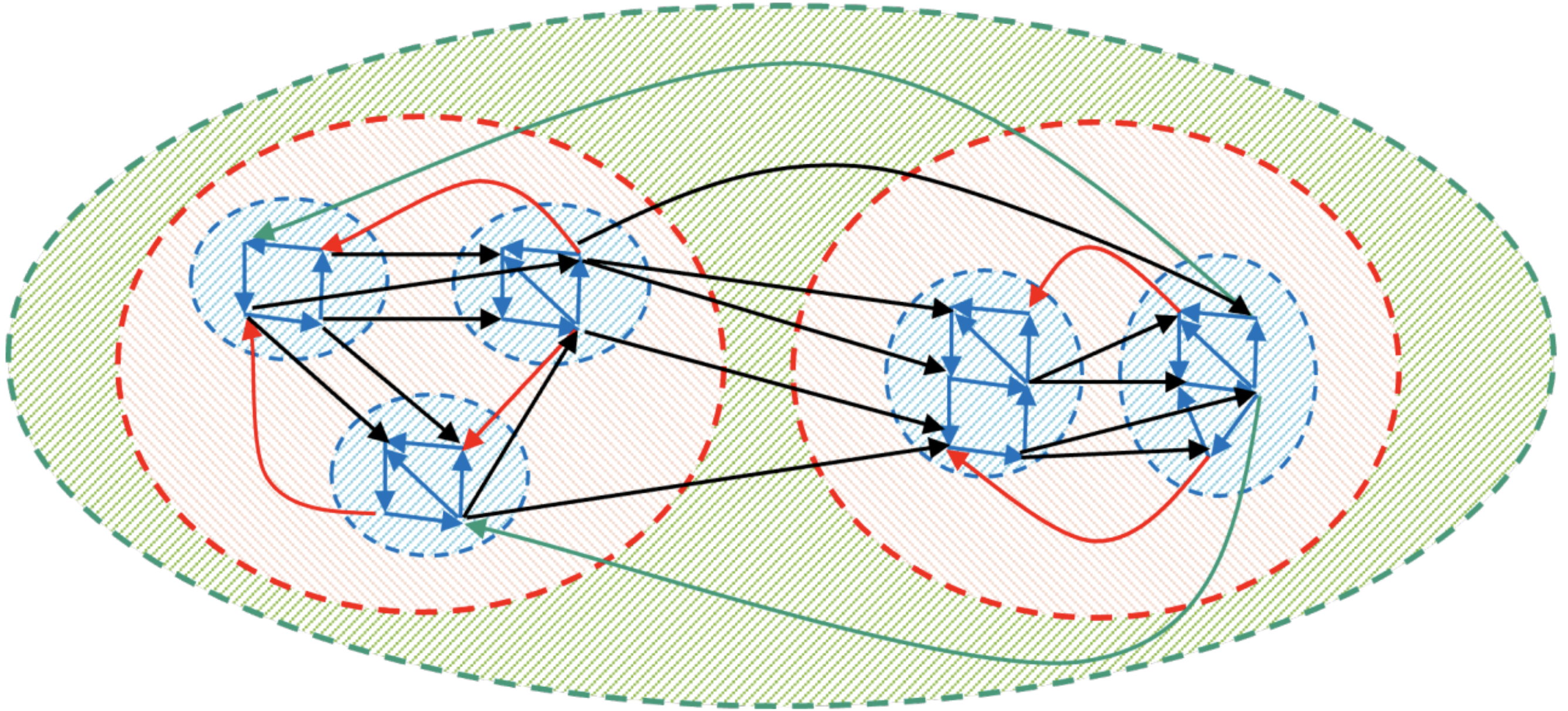
What to do about B ?

recurse!

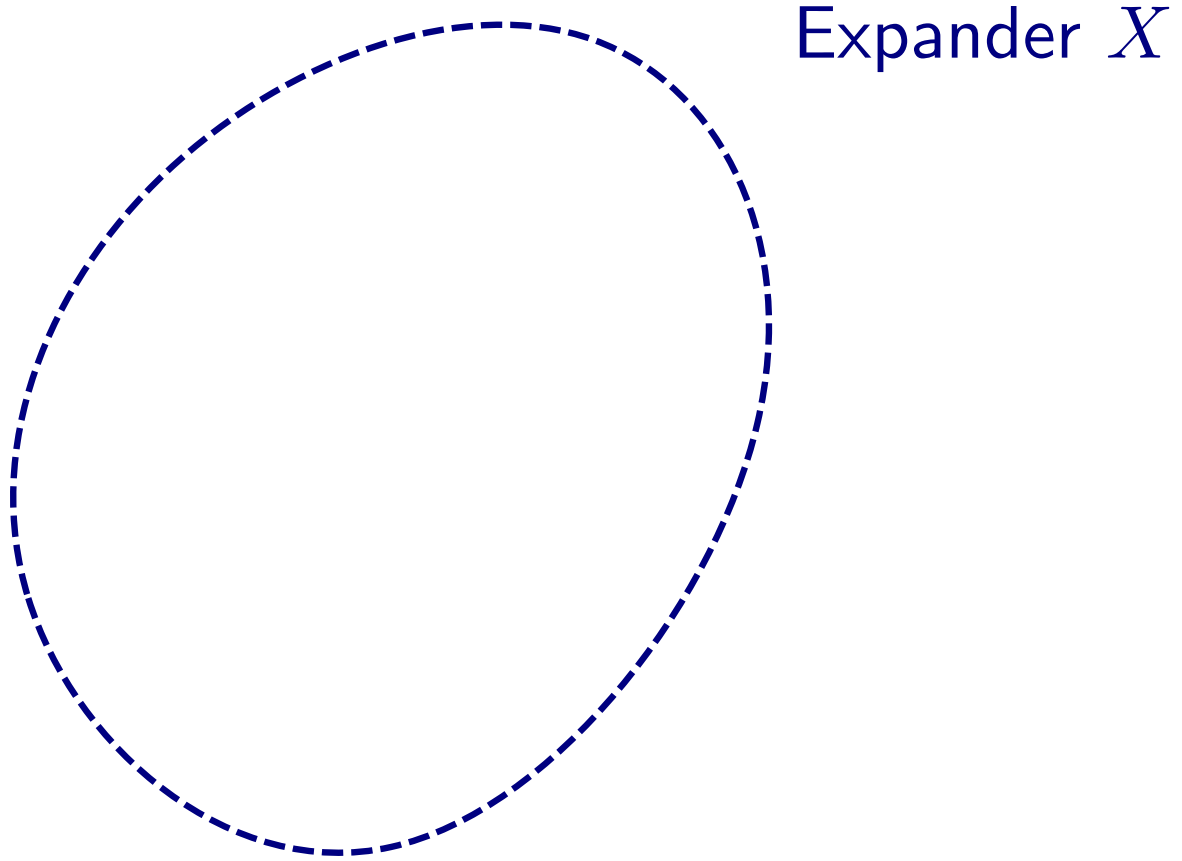
expander decomp.

w.r.t B

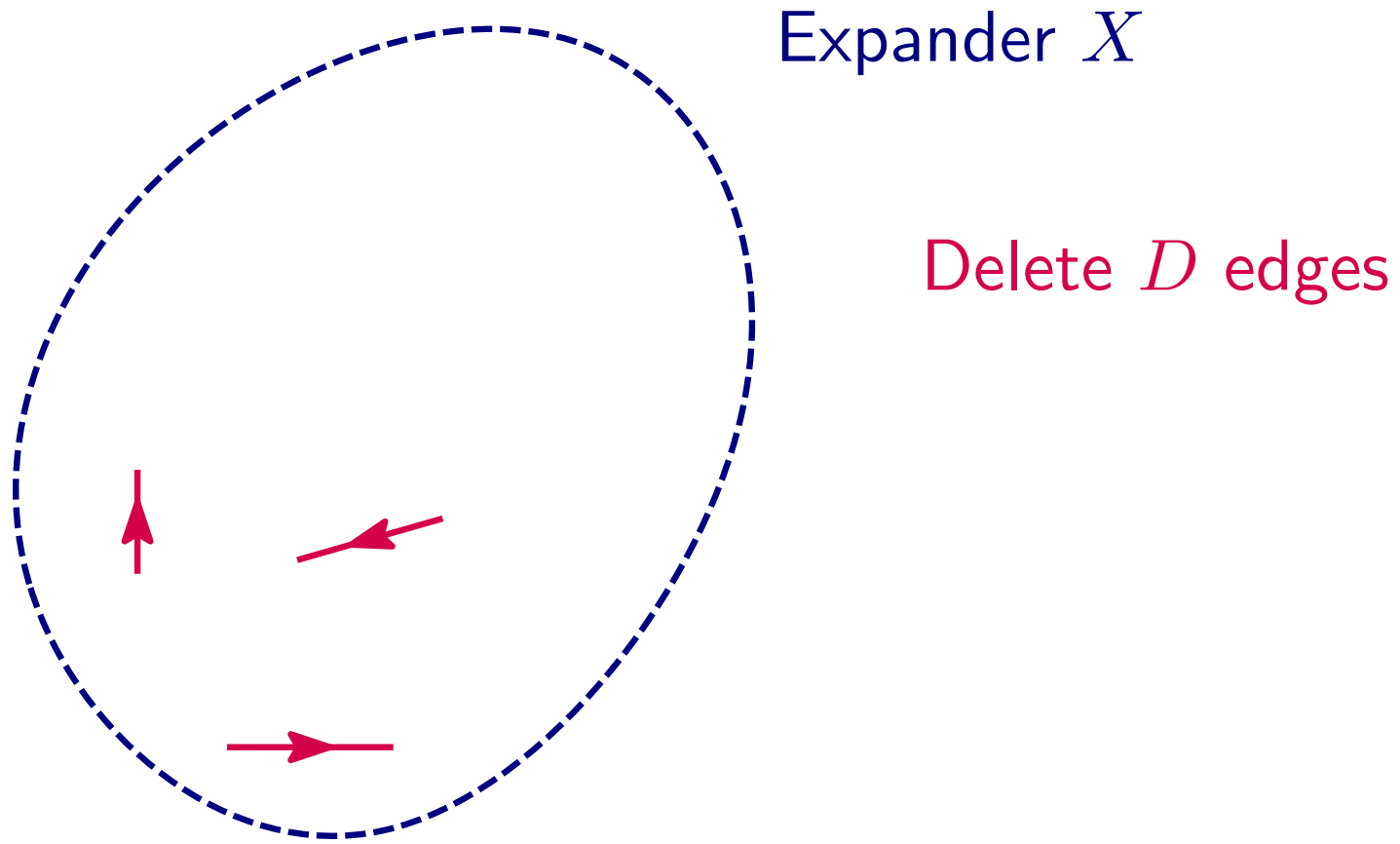
(Directed) Expander Decomposition



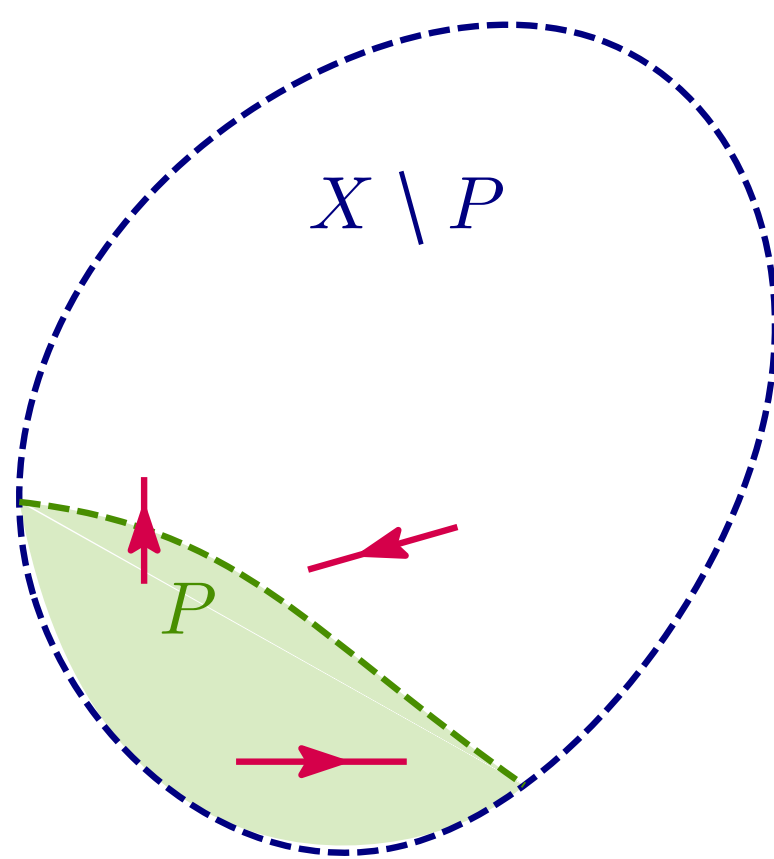
Technique Highlight: Path-Reversal Expander Pruning



Technique Highlight: Path-Reversal Expander Pruning



Technique Highlight: Path-Reversal Expander Pruning



Expander X

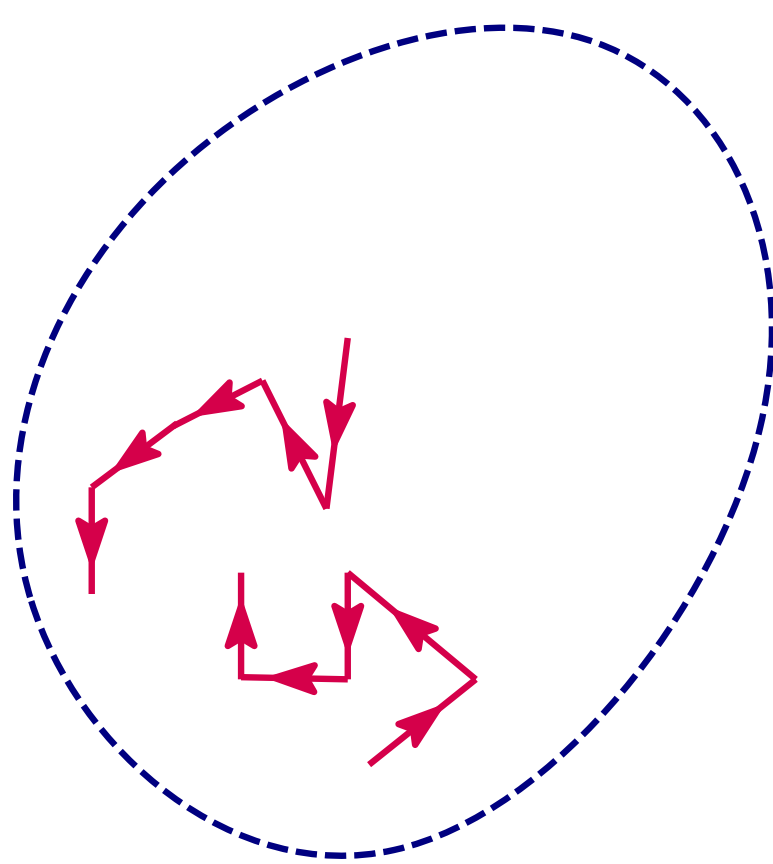
Delete D edges

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

$X \setminus P$ is still expander

Known: “Expander Pruning”

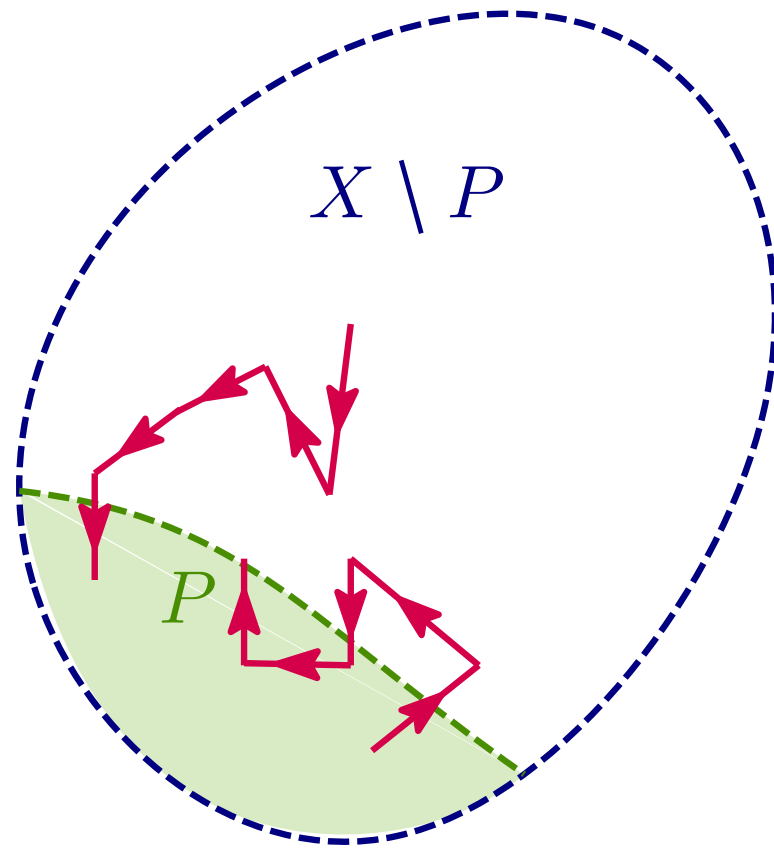
Technique Highlight: Path-Reversal Expander Pruning



Expander X

reverse D paths

Technique Highlight: Path-Reversal Expander Pruning



Expander X

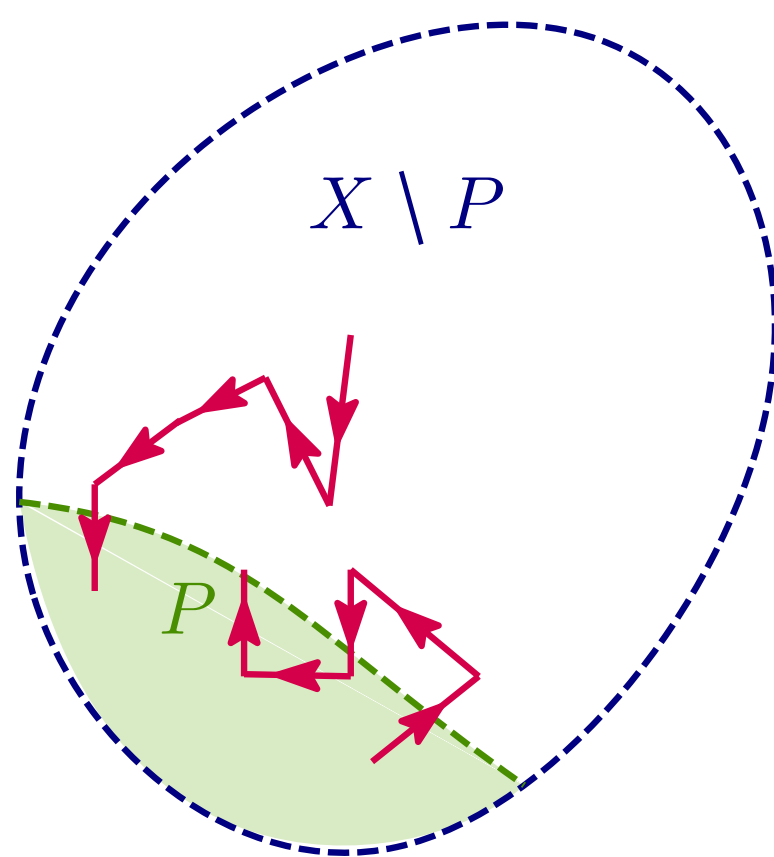
reverse D paths

Small "pruned" part P $\text{vol}(P) \leq 6|D|/\phi$

$X \setminus P$ is still expander

Theorem: Path-Reversal Expander Pruning

Technique Highlight: Path-Reversal Expander Pruning



Expander X

reverse D paths

Small "pruned" part P $\text{vol}(P) \leq 6|D|/\phi$

$X \setminus P$ is still expander

Theorem: Path-Reversal Expander Pruning

Directed Expander Hierarchy is robust under flow augmentation

Comparision

Ours

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

Comparision

Ours

Maximum Flow

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

Minimum Cost Maximum Flow

Comparision

Ours

Maximum Flow

$$\tilde{O}(n^2)$$

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

Minimum Cost Maximum Flow

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

Comparision

Ours

Maximum Flow

$$\tilde{O}(n^2)$$

Combinatorial
Augmenting Paths

Implementable

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

Minimum Cost Maximum Flow

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

Continuous Optimization
Dynamic Data Structures

Tricky to implement?