Can you link up with treewidth?

(to prove tight lower bounds for subgraph problems)

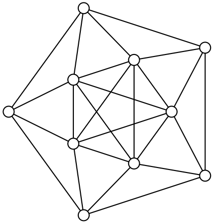
Jiaheng Wang *(Regensburg)*

with

Radu Curticapean *(ITU Copenhagen, Regensburg)*, Simon Döring *(Saarland)* and Daniel Neuen *(MPI)*

LFCS Seminar, 19 Nov 2024

A canonical problem

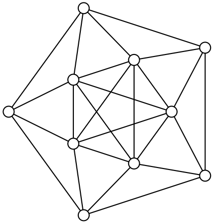


A canonical problem

CLIQUE

Input A graph G and an integer k

Output Does it have a size-k clique?

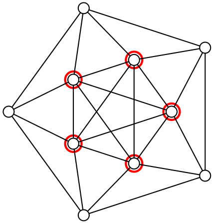


A canonical problem

CLIQUE

Input A graph G and an integer k

Output Does it have a size-k clique?

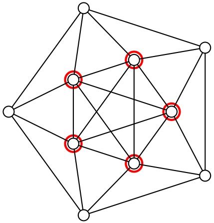


A canonically hard problem

CLIQUE

Input A graph G and an integer k

Output Does it have a size-k clique?



Theorem

[Karp'72]

CLIQUE is **NP**-complete.

A canonically hard problem

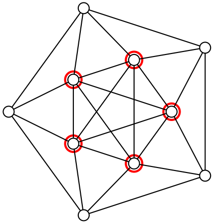
CLIQUE

Input A graph G and an integer k

Output Does it have a size-k clique?

~~poly(n) algorithm~~

unless **P** = **NP**



Theorem

[Karp'72]

CLIQUE is **NP**-complete.

A canonically hard parameterised problem

CLIQUE

Input A graph G and an integer k

Output Does it have a size-k clique?

pCLIQUE-k

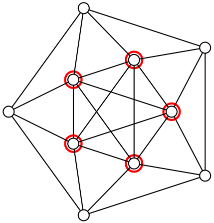
Input A graph G

Parameter An integer k

Output Does it have a size-k clique?

~~poly(n) algorithm~~

unless **P** = **NP**



Theorem [Karp'72]

CLIQUE is **NP**-complete.

A canonically hard parameterised problem

CLIQUE

Input A graph G and an integer k

Output Does it have a size-k clique?

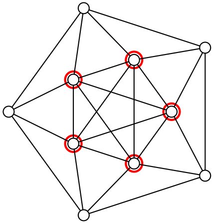
pCLIQUE-k

Input A graph G

Parameter An integer k

Output Does it have a size-k clique?

~~poly(n) algorithm~~
unless **P** = **NP**



Trivial $O(n^k)$ alg

Theorem [Karp'72]

CLIQUE is **NP**-complete.

A canonically hard parameterised problem

Clique

Input A graph G and an integer k

Output Does it have a size-k clique?

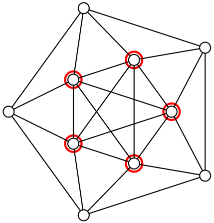
pClique-k

Input A graph G

Parameter An integer k

Output Does it have a size-k clique?

~~poly(n) algorithm~~
unless **P** = **NP**



Trivial $O(n^k)$ alg

Theorem [Karp'72]

Clique is **NP**-complete.

Theorem [Downey–Fellows'95]

pClique is **W[1]**-complete.

A canonically hard parameterised problem

CLIQUE

Input A graph G and an integer k

Output Does it have a size-k clique?

pCLIQUE-k

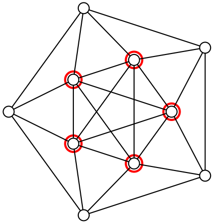
Input A graph G

Parameter An integer k

Output Does it have a size-k clique?

~~poly(n) algorithm~~

unless **P** = **NP**



Trivial $O(n^k)$ alg

~~$f(k) \cdot \text{poly}(n)$ alg~~

unless **FPT** = **W[1]**

Theorem [Karp'72]

CLIQUE is **NP**-complete.

Theorem [Downey–Fellows'95]

pCLIQUE is **W[1]**-complete.

A canonically hard parameterised problem

Clique

Input A graph G and an integer k

Output Does it have a size-k clique?

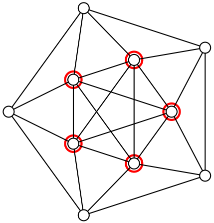
pClique-k

Input A graph G

Parameter An integer k

Output Does it have a size-k clique?

$n^{\log n}$
$2^{\sqrt{n}}$ algorithm?



Trivial $O(n^k)$ alg

~~$f(k) \cdot \mathrm{poly}(n)$ alg~~
unless **FPT** = **W**[1]

Theorem [Karp'72]

Clique is **NP**-complete.

Theorem [Downey–Fellows'95]

pClique is **W**[1]-complete.

A canonically hard parameterised problem

CLIQUE

Input A graph G and an integer k

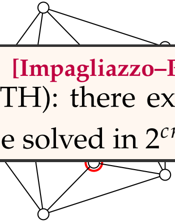
Output Does it have a size-k clique?

pCLIQUE-k

Input A graph G

Parameter An integer k

Output Does it have a size-k clique?



Trivial $O(n^k)$ alg.

unless **FPT** = **W[1]**

Conjecture

[Impagliazzo–Paturi'99, Impagliazzo–Paturi–Zane'01]

Exponential-Time Hypothesis (ETH): there exists an absolute constant c such that 3-SAT on n variables cannot be solved in 2^{cn} time.

Theorem

[Karp'72]

CLIQUE is **NP**-complete.

Theorem

[Downey–Fellows'95]

pCLIQUE is **W[1]**-complete.

A canonically hard parameterised problem

Clique

Input A graph G and an integer k

Output Does it have a size-k clique?

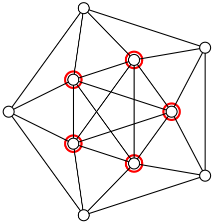
pClique-k

Input A graph G

Parameter An integer k

Output Does it have a size-k clique?

~~$2^{o(n)}$ algorithm~~
unless ETH fails



Trivial $O(n^k)$ alg

~~$f(k) \cdot \mathrm{poly}(n)$ alg~~
unless **FPT** = **W**[1]

Theorem [Karp'72]

Clique is **NP**-complete.

Theorem [Downey–Fellows'95]

pClique is **W**[1]-complete.

A canonically hard parameterised problem

CLIQUE

Input A graph G and an integer k

Output Does it have a size-k clique?

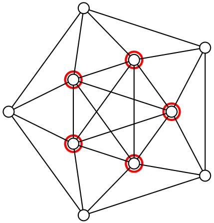
pCLIQUE-k

Input A graph G

Parameter An integer k

Output Does it have a size-k clique?

$\cancel{2^{o(n)} \text{ algorithm}}$
unless ETH fails



Trivial $O(n^k)$ alg

$f(k) \cdot n^{\sqrt{k}}$ alg?

Theorem [Karp'72]

CLIQUE is **NP**-complete.

Theorem [Downey–Fellows'95]

pCLIQUE is **W[1]**-complete.

A fine-grained canonically hard parameterised problem

CLIQUE

Input A graph G and an integer k

Output Does it have a size-k clique?

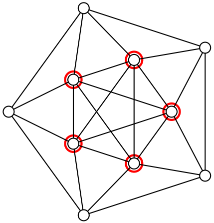
pCLIQUE-k

Input A graph G

Parameter An integer k

Output Does it have a size-k clique?

$\cancel{2^{o(n)}}$ ~~algorithm~~
unless ETH fails



Trivial $O(n^k)$ alg

$\cancel{f(k) \cdot n^{o(k)}}$ ~~alg~~
unless ETH fails

Theorem

[Chen–Huang–Kanj–Xia'06]

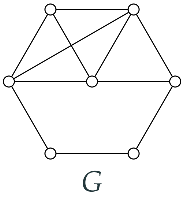
pCLIQUE-k cannot be solved in time $f(k)n^{o(k)}$ unless ETH fails.

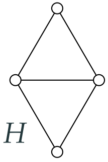
Subgraph detection problem

SUB(H)

Input A graph G

Output Does it have a subgraph (isomorphic to) H?



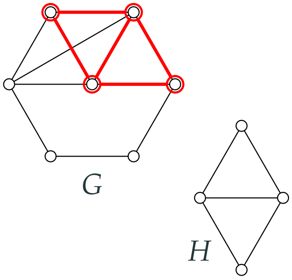


Subgraph detection problem

SUB(H)

Input A graph G

Output Does it have a subgraph (isomorphic to) H?

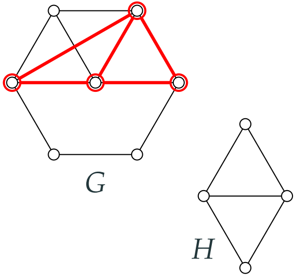


Subgraph detection problem

SUB(H)

Input A graph G

Output Does it have a subgraph (isomorphic to) H?

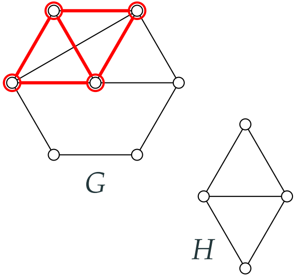


Subgraph detection problem

SUB(H)

Input A graph G

Output Does it have a subgraph (isomorphic to) H?

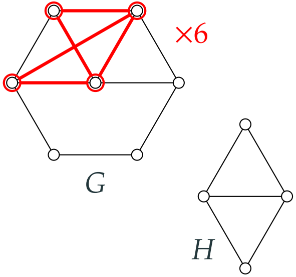


Subgraph detection problem

Sub(H)

Input A graph G

Output Does it have a subgraph (isomorphic to) H?



Subgraph detection problem

SUB(H)

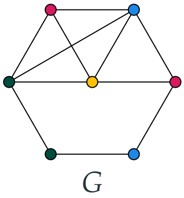
Input A graph G

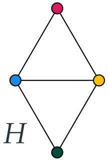
Output Does it have a subgraph (isomorphic to) H?

COLSUB(H)

Input A vertex-coloured graph G

Output Does it have a colourful copy of H?





Subgraph detection problem

Sub(H)

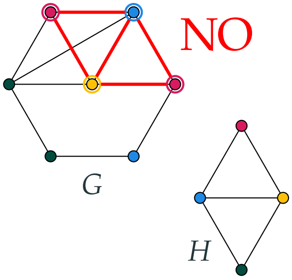
Input A graph G

Output Does it have a subgraph (isomorphic to) H?

ColSub(H)

Input A vertex-coloured graph G

Output Does it have a colourful copy of H?



Subgraph detection problem

SUB(H)

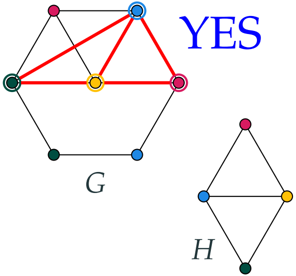
Input A graph G

Output Does it have a subgraph (isomorphic to) H?

COLSUB(H)

Input A vertex-coloured graph G

Output Does it have a colourful copy of H?



SUB(H)

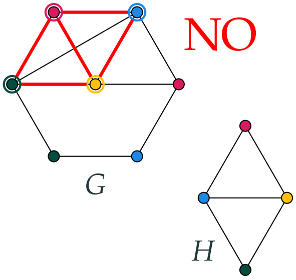
Input A graph G

Output Does it have a subgraph (isomorphic to) H?

COLSUB(H)

Input A vertex-coloured graph G

Output Does it have a colourful copy of H?



SUB(H)

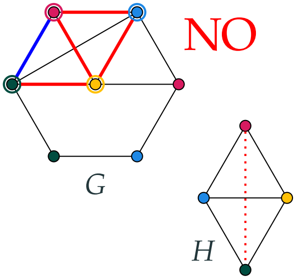
Input A graph G

Output Does it have a subgraph (isomorphic to) H?

COLSUB(H)

Input A vertex-coloured graph G

Output Does it have a colourful copy of H?



Subgraph detection problem

Sub(H)

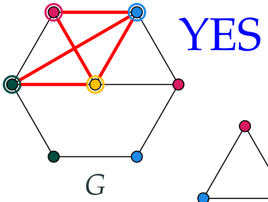
Input A graph G

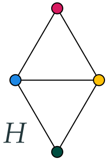
Output Does it have a subgraph (isomorphic to) H?

ColSub(H)

Input A vertex-coloured graph G

Output Does it have a colourful copy of H?





Subgraph detection problem

$\#\textsc{Sub}(H)$

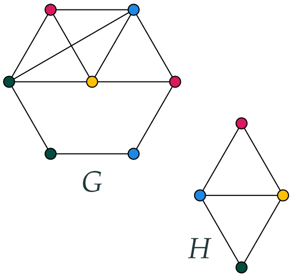
Input A graph G

Output Number of subgraphs (isomorphic to) H

$\#\textsc{ColSub}(H)$

Input A vertex-coloured graph G

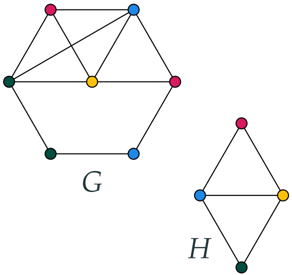
Output Number of colourful copies of H



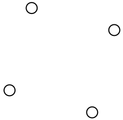
ColSub(H)

Input A vertex-coloured graph G

Output Does it have a colourful copy of H?



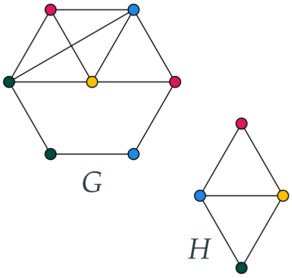
Indset



ColSub(H)

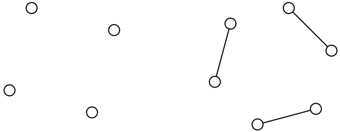
Input A vertex-coloured graph G

Output Does it have a colourful copy of H?



Indset

Matching

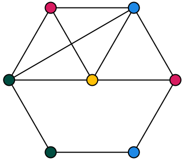


Subgraph detection problem

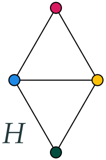
COLSUB(H)

Input A vertex-coloured graph G

Output Does it have a colourful copy of H?

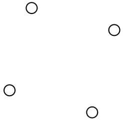


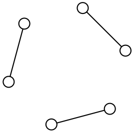
G

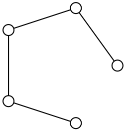


H

Indset	Matching	Path





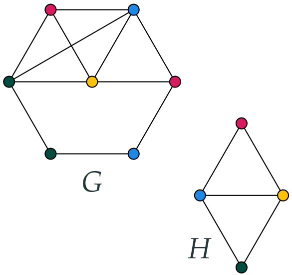


Subgraph detection problem

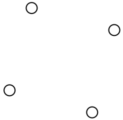
COLSUB(H)

Input A vertex-coloured graph G

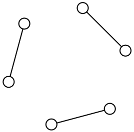
Output Does it have a colourful copy of H?



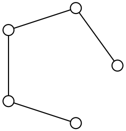
Indset



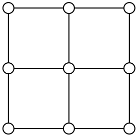
Matching



Path



Grid

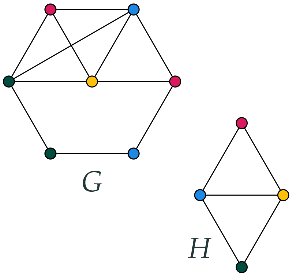


Subgraph detection problem

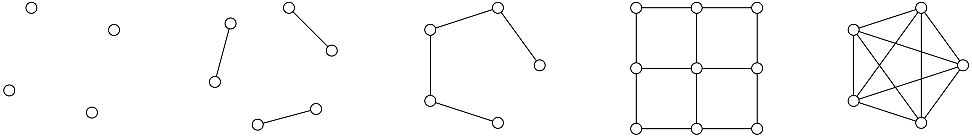
ColSub(H)

Input A vertex-coloured graph G

Output Does it have a colourful copy of H?



Indset	Matching	Path	Grid	Clique

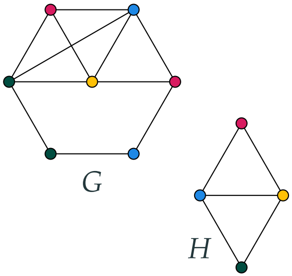


Subgraph detection problem

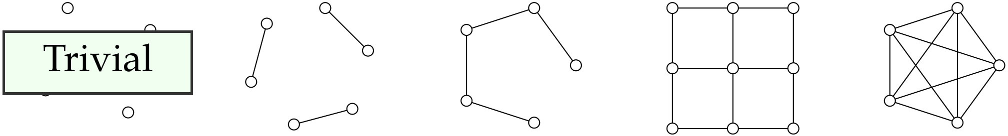
ColSub(H)

Input A vertex-coloured graph G

Output Does it have a colourful copy of H?



Indset	Matching	Path	Grid	Clique

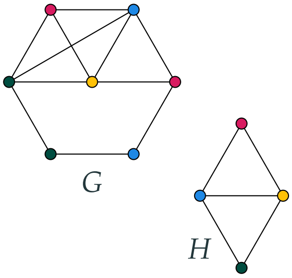


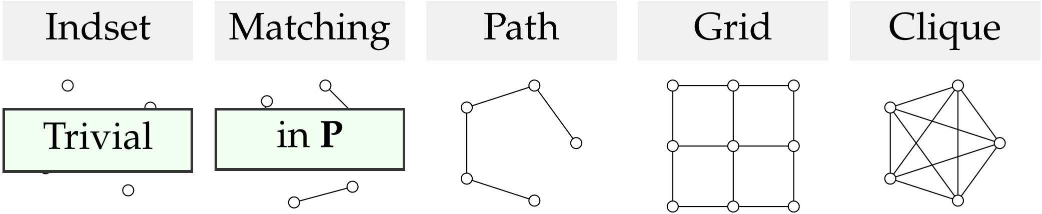
Subgraph detection problem

ColSub(H)

Input A vertex-coloured graph G

Output Does it have a colourful copy of H?



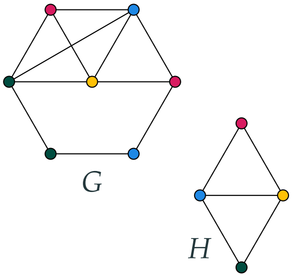


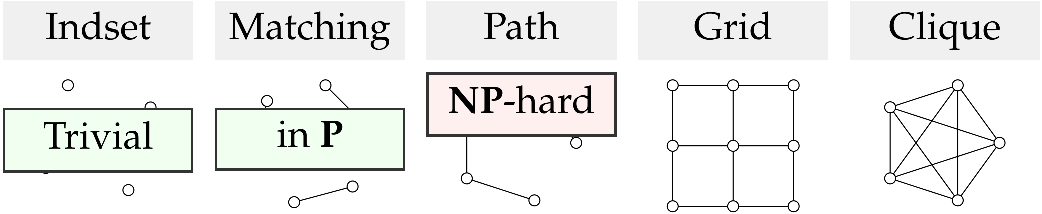
Subgraph detection problem

ColSub(H)

Input A vertex-coloured graph G

Output Does it have a colourful copy of H?



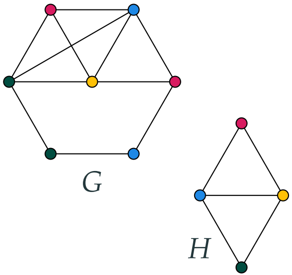


Subgraph detection problem

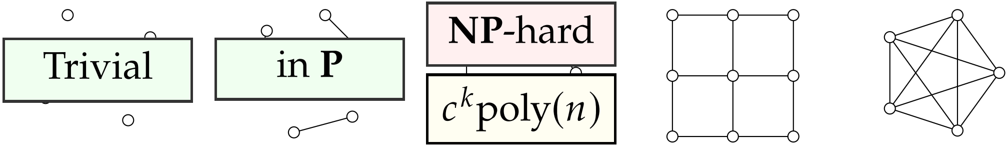
ColSub(H)

Input A vertex-coloured graph G

Output Does it have a colourful copy of H?



Indset	Matching	Path	Grid	Clique
Trivial	in **P**	**NP**-hard / $c^k\text{poly}(n)$		

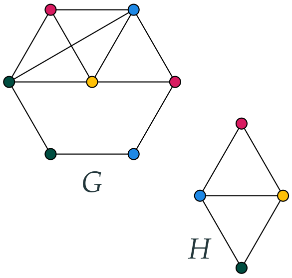


Subgraph detection problem

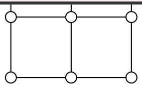
ColSub(H)

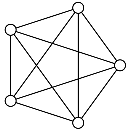
Input A vertex-coloured graph G

Output Does it have a colourful copy of H?



Indset	Matching	Path	Grid	Clique
Trivial	in **P**	**NP**-hard $c^k \text{poly}(n)$	**W[1]**-hard	



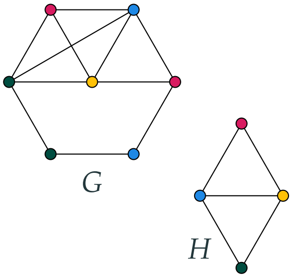


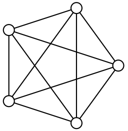
Subgraph detection problem

COLSUB(H)

Input A vertex-coloured graph G

Output Does it have a colourful copy of H?



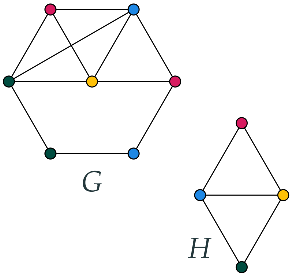
Indset	Matching	Path	Grid	Clique
Trivial	in **P**	**NP**-hard	**W[1]**-hard	
		$c^k\text{poly}(n)$	$n^{\sqrt{k}}$	

Subgraph detection problem

COLSUB(H)

Input A vertex-coloured graph G

Output Does it have a colourful copy of H?

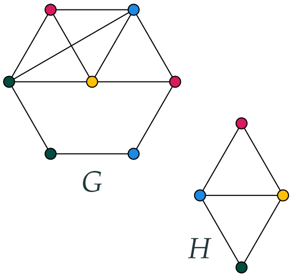


Indset	Matching	Path	Grid	Clique
Trivial	in **P**	**NP**-hard	**W[1]**-hard	**W[1]**-hard
		$c^k \mathrm{poly}(n)$	$n^{\sqrt{k}}$	

ColSub(H)

Input A vertex-coloured graph G

Output Does it have a colourful copy of H?

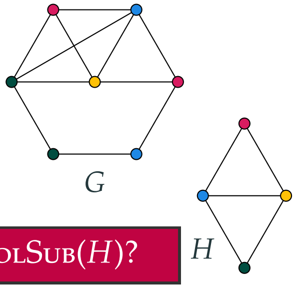


Indset	Matching	Path	Grid	Clique
Trivial	in **P**	**NP**-hard	**W[1]**-hard	**W[1]**-hard
		$c^k \mathrm{poly}(n)$	$n^{\sqrt{k}}$	'ETH-hard'

ColSub(H)

Input A vertex-coloured graph G

Output Does it have a colourful copy of H?



How can we prove tight lower bound for ColSub(H)?

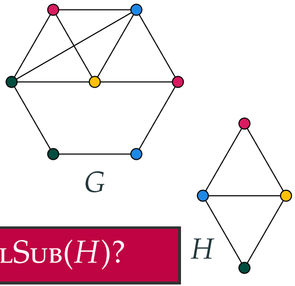
Indset	Matching	Path	Grid	Clique
Trivial	in **P**	**NP**-hard	**W[1]**-hard	**W[1]**-hard
		$c^k \mathrm{poly}(n)$	$n^{\sqrt{k}}$	'ETH-hard'

Subgraph detection problem

ColSub(H)

Input A vertex-coloured graph G

Output Does it have a colourful copy of H?



Why **do we need tight lower bound for** ColSub(H)?

Indset	Matching	Path	Grid	Clique
Trivial	in **P**	**NP**-hard	**W[1]**-hard	**W[1]**-hard
		$c^k \mathrm{poly}(n)$	$n^{\sqrt{k}}$	'ETH-hard'

Is pClique a good source of hardness?

Theorem

[Chen–Huang–Kanj–Xia'06]

pClique-k cannot be solved in time $f(k)n^{o(k)}$ unless ETH fails.

Is pClique a good source of hardness?

Theorem

[Chen–Huang–Kanj–Xia'06]

pClique-k cannot be solved in time $f(k)n^{o(k)}$ unless ETH fails.

Yes it is, but

Is pClique a good source of hardness?

Theorem

[Chen–Huang–Kanj–Xia'06]

pClique-k cannot be solved in time $f(k)n^{o(k)}$ unless ETH fails.

Yes it is, but it has $\Theta(k^2)$ edges.

Is pClique a good source of hardness?

Theorem

[Chen–Huang–Kanj–Xia'06]

pClique-k cannot be solved in time $f(k)n^{o(k)}$ unless ETH fails.

Yes it is, but it has $\Theta(k^2)$ edges.

Is pClique a good source of hardness?

Theorem

[Chen–Huang–Kanj–Xia'06]

pClique-k cannot be solved in time $f(k)n^{o(k)}$ unless ETH fails.

Yes it is, but it has $\Theta(k^2)$ edges.

I only need
W[1]-hardness!

Is pClique a good source of hardness?

Theorem [Chen–Huang–Kanj–Xia'06]

pClique-k cannot be solved in time $f(k)n^{o(k)}$ unless ETH fails.

Yes it is, but it has $\Theta(k^2)$ edges.

I only need **W[1]**-hardness!

Is pClique a good source of hardness?

Theorem [Chen–Huang–Kanj–Xia'06]

pClique-k cannot be solved in time $f(k)n^{o(k)}$ unless ETH fails.

Yes it is, but it has $\Theta(k^2)$ edges.

I only need
W[1]-hardness!

Hmm... I want $n^{\Omega(t)}$...
But this only gives me $n^{\Omega(\sqrt{t})}$!

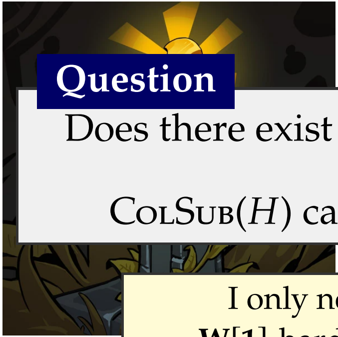
Is pClique a good source of hardness?

Theorem

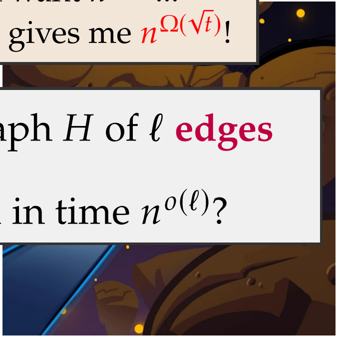
[Chen–Huang–Kanj–Xia'06]

pClique-k cannot be solved in time $f(k)n^{o(k)}$ unless ETH fails.

Yes it is, but it has $\Theta(k^2)$ edges.



Hmm... I want $n^{\Omega(t)}$...
But this only gives me $n^{\Omega(\sqrt{t})}$!



Question

Does there exist any **sparse** graph H of ℓ **edges** such that ColSub(H) cannot be solved in time $n^{o(\ell)}$?

I only need **W[1]**-hardness!

Hardness of sparse subgraphs

Question

Does there exist any **sparse** graph H of ℓ **edges** such that $\textsc{ColSub}(H)$ cannot be solved in time $n^{o(\ell)}$?

Question

Does there exist any **sparse** graph H of ℓ **edges** such that $\textsc{ColSub}(H)$ cannot be solved in time $n^{o(\ell)}$?

If this is true, then we have **tight** lower bounds for these parameterised problems/algorithms:

Question

Does there exist any **sparse** graph H of ℓ **edges** such that $\textsc{ColSub}(H)$ cannot be solved in time $n^{o(\ell)}$?

If this is true, then we have **tight** lower bounds for these parameterised problems/algorithms:

Feedback edge set [Agrawal–Panolan–Saurabh'21]
Bounded congestion routing [Amiri–Kreutzer–Marx'19]
Red-blue points separation [Bonnet–Giannopoulos–Lampis'19]
k-OPT in TSP [Bonnet–Iwata–Jansen–Kowalik'19]
Art gallery [Bonnet–Miltzow'20]

Question

Does there exist any **sparse** graph H of ℓ **edges** such that $\textsc{ColSub}(H)$ cannot be solved in time $n^{o(\ell)}$?

If this is true, then we have **tight** lower bounds for these parameterised problems/algorithms:

Feedback edge set **[Agrawal–Panolan–Saurabh'21]**
Bounded congestion routing **[Amiri–Kreutzer–Marx'19]**
Red-blue points separation **[Bonnet–Giannopoulos–Lampis'19]**
k-OPT in TSP **[Bonnet–Iwata–Jansen–Kowalik'19]**
Art gallery **[Bonnet–Miltzow'20]**

Unfortunately, we don't know if there is any such H!

Question

Does there exist any **sparse** graph H of ℓ **edges** such that $\textsc{ColSub}(H)$ cannot be solved in time $n^{o(\ell)}$?

If this is true, then we have **tight** lower bounds for these parameterised problems/algorithms:

Feedback edge set **[Agrawal–Panolan–Saurabh'21]**
Bounded congestion routing **[Amiri–Kreutzer–Marx'19]**
Red-blue points separation **[Bonnet–Giannopoulos–Lampis'19]**
k-OPT in TSP **[Bonnet–Iwata–Jansen–Kowalik'19]**
Art gallery **[Bonnet–Miltzow'20]**

Unfortunately, we don't know if there is any such H! However...

Almost tight hard instances

Theorem

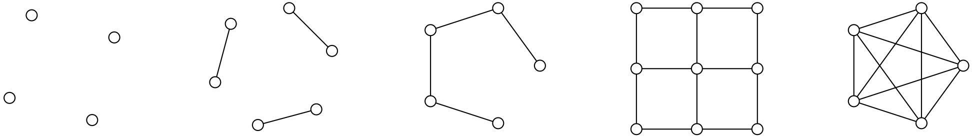
[Marx'10]

There is a sequence of **degree-3** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and ColSub(H_ℓ) cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.

Theorem

[Marx'10]

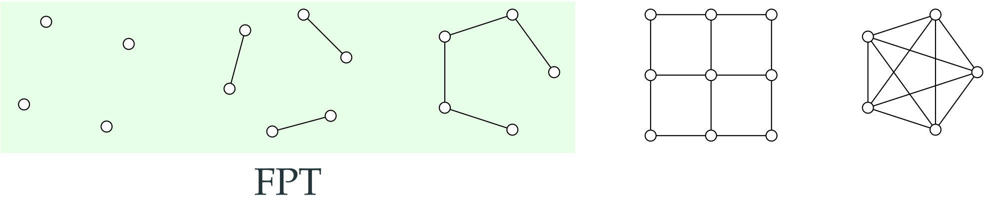
There is a sequence of **degree-3** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and ColSub(H_ℓ) cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.



Theorem

[Marx'10]

There is a sequence of **degree-3** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and ColSub(H_ℓ) cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.

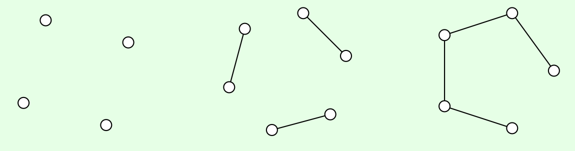


Almost tight hard instances

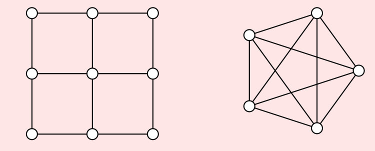
Theorem

[Marx'10]

There is a sequence of **degree-3** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and ColSub(H_ℓ) cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.



FPT



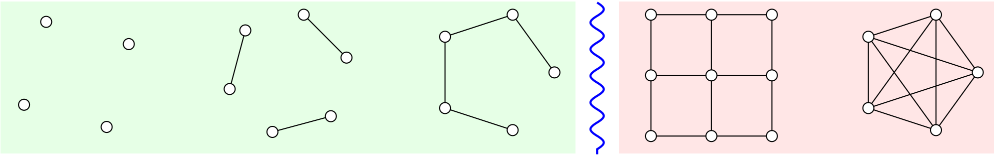
W[1]-hard

Almost tight hard instances

Theorem

[Marx'10]

There is a sequence of **degree-3** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and ColSub(H_ℓ) cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.

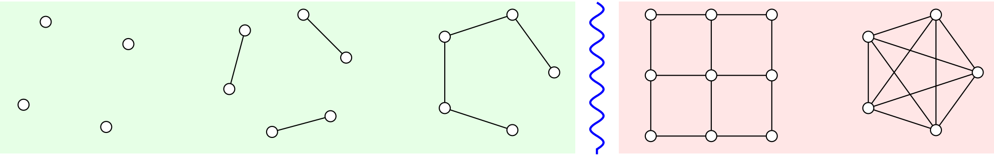


FPT ? **W[1]**-hard

Theorem

[Marx'10]

There is a sequence of **degree-3** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and $\textsc{ColSub}(H_\ell)$ cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.



FPT ? **W[1]**-hard

Theorem

[Grohe–Schwentick–Segoufin'01, Grohe'07]

$\textsc{ColSub}(\mathcal{H})$ is tractable if and only if $\mathcal{H}$ is a bounded-**treewidth** graph family, unless **FPT** = **W[1]**.

Almost tight hard instances

Theorem

[Marx'10]

There is a sequence of **degree-3** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and ColSub(H_ℓ) cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.

THEORY OF COMPUTING, Volume 6 (2010), pp. 85–112
www.theoryofcomputing.org

Can You Beat Treewidth?*

Dániel Marx†

Received: September 3, 2008; published: August 27, 2010.

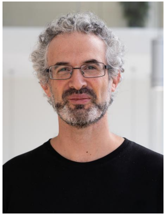
Theorem

[Marx'10]

There is a sequence of **degree-3** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and $\textsc{ColSub}(H_\ell)$ cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.

Can You Beat Treewidth?

No, you can't. Treewidth is *almost* optimal.



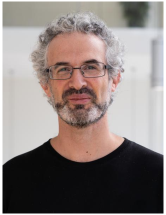
Theorem

[Marx'10]

There is a sequence of **degree-3** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and $\textsc{ColSub}(H_\ell)$ cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.

Can You Beat Treewidth?

No, you can't. Treewidth is *almost* optimal.



Theorem

[Marx'10]

Let $\mathcal{H}$ be **any** unbounded-treewidth graph family.
Then $\textsc{ColSub}(H)$ for $H \in \mathcal{H}$ cannot be solved in time $f(H)n^{o(\mathrm{tw}(H)/\log \mathrm{tw}(H))}$ unless ETH fails.

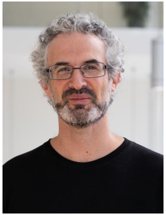
Theorem

[Marx'10]

There is a sequence of **degree-3** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and $\textsc{ColSub}(H_\ell)$ cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.

Can You Beat Treewidth?

No, you can't. Treewidth is *almost* optimal.



Theorem

[Too many papers]

There is an explicit construction of degree-3 **expanders**.

Theorem

[Grohe–Marx'08]

Expanders has linear treewidth.

Folklore corollary of ETH

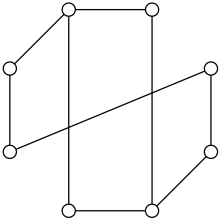
$\exists c > 0$ s.t. 3-Colouring cannot be solved in time $O(2^{cn})$ on degree-4 graphs.

$$\text{3-Colouring}[2^{c_1 n}] \quad \preceq \quad \text{pClique-}k[N^{c_2 k}]$$

Folklore corollary of ETH

$\exists c > 0$ s.t. 3-Colouring cannot be solved in time $O(2^{cn})$ on degree-4 graphs.

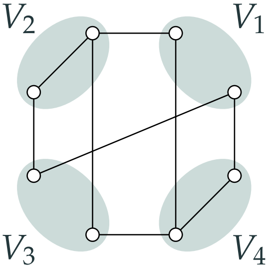
$$3\text{-Colouring}[2^{c_1 n}] \quad \leq \quad \text{pClique-}k[N^{c_2 k}]$$



Folklore corollary of ETH

$\exists c > 0$ s.t. 3-Colouring cannot be solved in time $O(2^{cn})$ on degree-4 graphs.

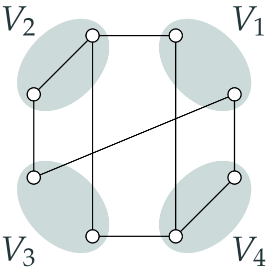
$$\text{3-Colouring}[2^{c_1 n}] \quad \preceq \quad \text{pClique-}k[N^{c_2 k}]$$

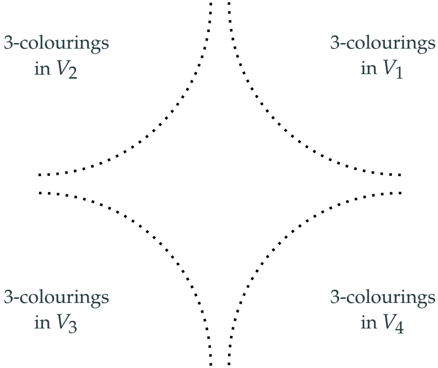


Folklore corollary of ETH

$\exists c > 0$ s.t. 3-COLOURING cannot be solved in time $O(2^{cn})$ on degree-4 graphs.

$$\text{3-COLOURING}[2^{c_1 n}] \quad \preceq \quad \text{pCLIQUE-}k[N^{c_2 k}]$$

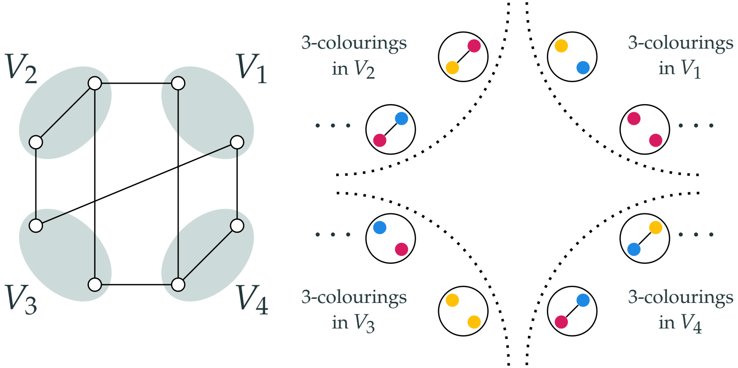




Folklore corollary of ETH

$\exists c > 0$ s.t. 3-Colouring cannot be solved in time $O(2^{cn})$ on degree-4 graphs.

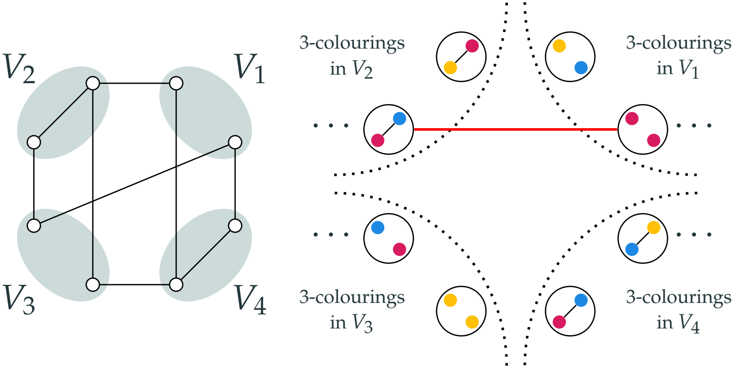
$$\text{3-Colouring}[2^{c_1 n}] \quad \preceq \quad \text{pClique-}k[N^{c_2 k}]$$



Folklore corollary of ETH

$\exists c > 0$ s.t. 3-COLOURING cannot be solved in time $O(2^{cn})$ on degree-4 graphs.

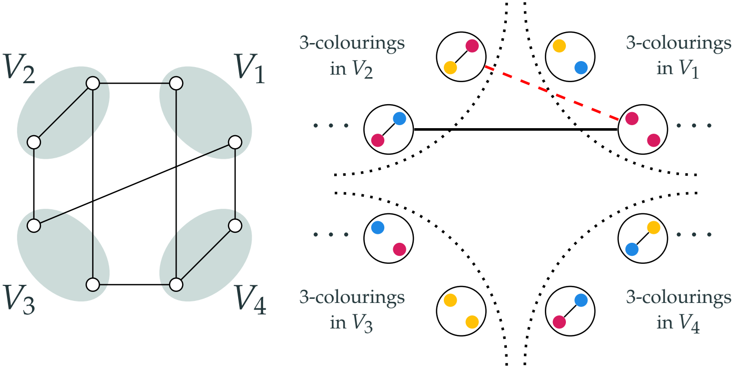
$$\text{3-COLOURING}[2^{c_1 n}] \quad \leq \quad \text{pCLIQUE-}k[N^{c_2 k}]$$



Folklore corollary of ETH

$\exists c > 0$ s.t. 3-COLOURING cannot be solved in time $O(2^{cn})$ on degree-4 graphs.

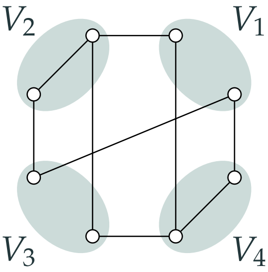
$$\text{3-COLOURING}[2^{c_1 n}] \quad \preceq \quad \text{pCLIQUE-}k[N^{c_2 k}]$$

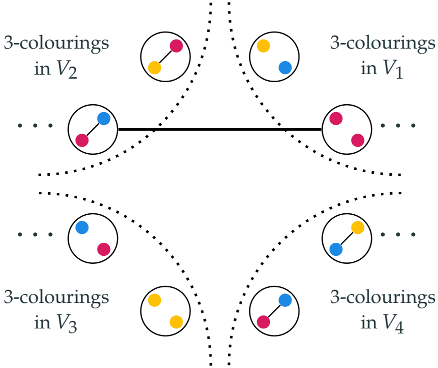


Folklore corollary of ETH

$\exists c > 0$ s.t. 3-COLOURING cannot be solved in time $O(2^{cn})$ on degree-4 graphs.

$$\text{3-COLOURING}[2^{c_1 n}] \quad \preceq \quad \text{pCLIQUE-}k[N^{c_2 k}]$$

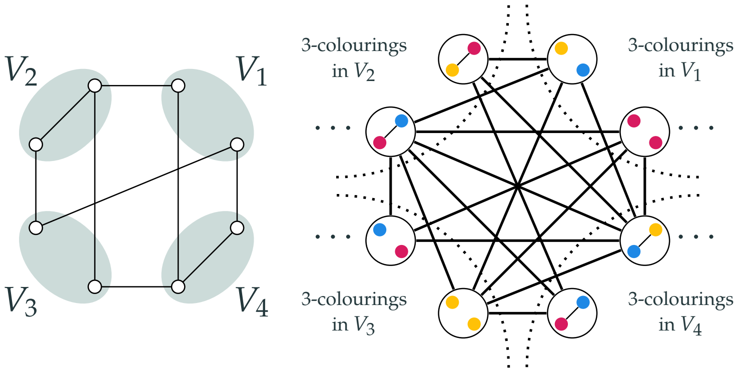




Folklore corollary of ETH

$\exists c > 0$ s.t. 3-Colouring cannot be solved in time $O(2^{cn})$ on degree-4 graphs.

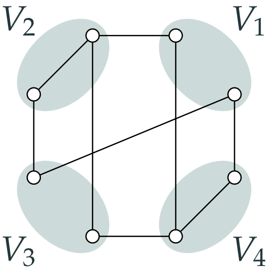
$$\text{3-Colouring}[2^{c_1 n}] \quad \preceq \quad \text{pClique-}k[N^{c_2 k}]$$

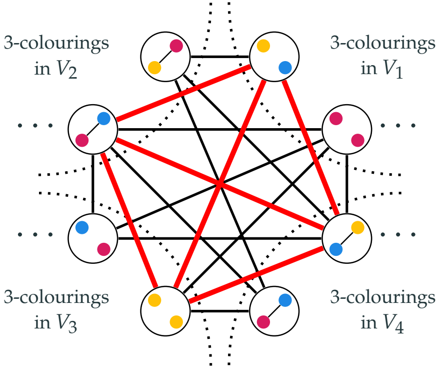


Folklore corollary of ETH

$\exists c > 0$ s.t. 3-Colouring cannot be solved in time $O(2^{cn})$ on degree-4 graphs.

$$\text{3-Colouring}[2^{c_1 n}] \quad \leq \quad \text{pClique-}k[N^{c_2 k}]$$

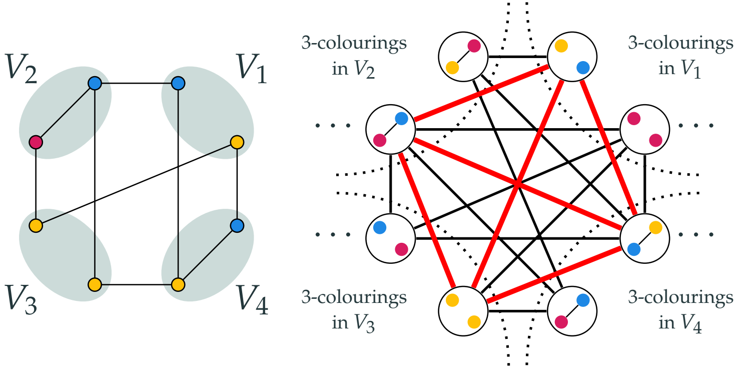




Folklore corollary of ETH

$\exists c > 0$ s.t. 3-Colouring cannot be solved in time $O(2^{cn})$ on degree-4 graphs.

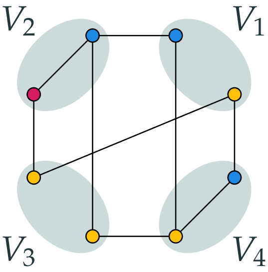
$$\text{3-Colouring}[2^{c_1 n}] \quad \leq \quad \text{pClique-}k[N^{c_2 k}]$$

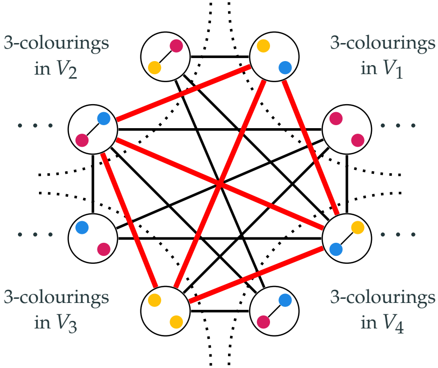


Folklore corollary of ETH

$\exists c > 0$ s.t. 3-Colouring cannot be solved in time $O(2^{cn})$ on degree-4 graphs.

$$\#\text{3-Colouring}(G) \quad = \quad \#\text{pClique-}k(G^*)$$





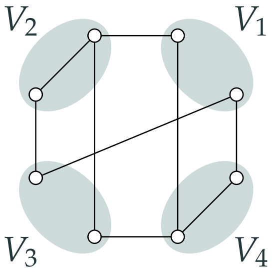
Folklore corollary of ETH

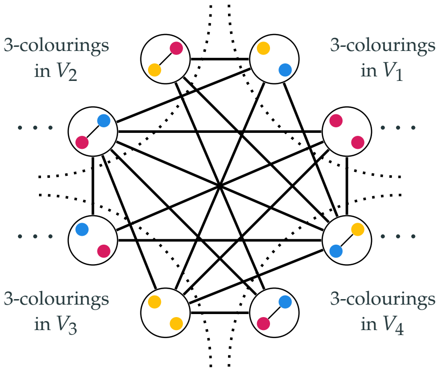
$\exists c > 0$ s.t. 3-Colouring cannot be solved in time $O(2^{cn})$ on degree-4 graphs.

$$\text{3-Colouring}[2^{c_1 n}] \quad \preceq \quad \text{pClique-}k[N^{c_2 k}]$$

n vertices

k parts

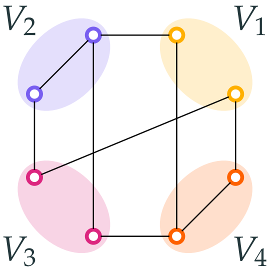


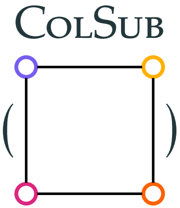


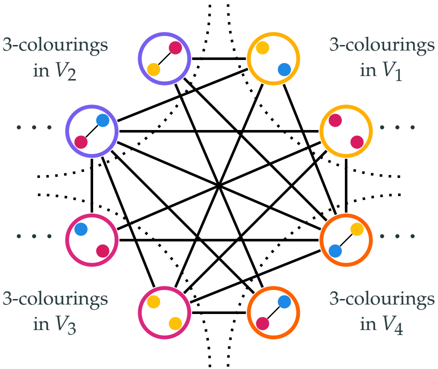
$N = k \cdot 3^{n/k}$ vertices

$$N^{c_2 k} = k^{c_2 k} 2^{(c_2 \log 3/\log 2)\cdot n}$$

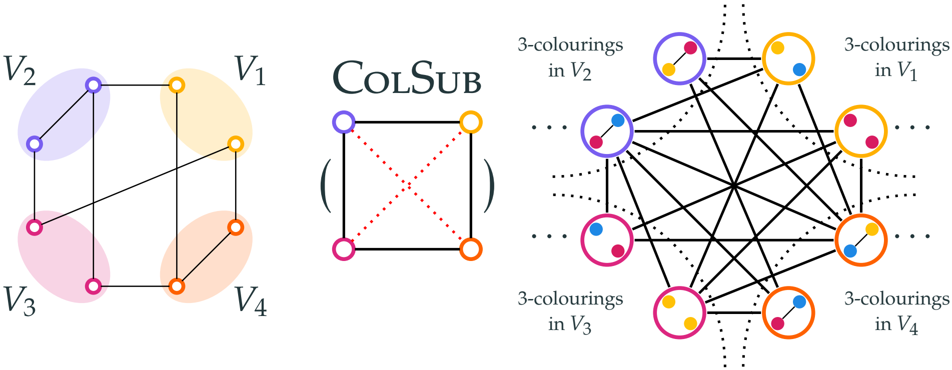
A framework of reduction: general patterns



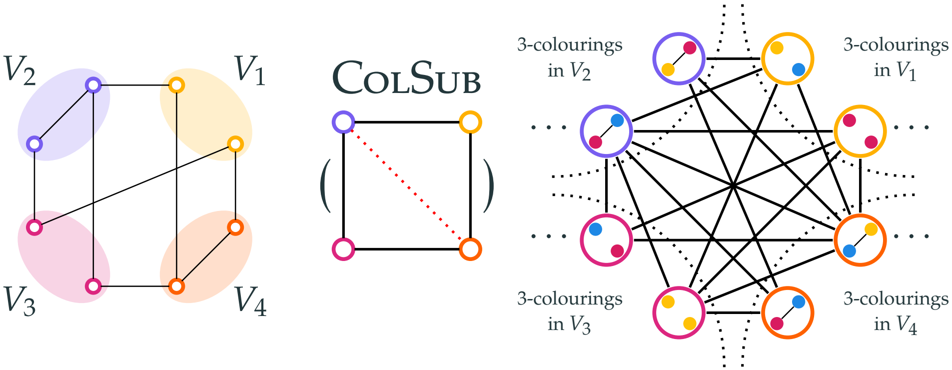




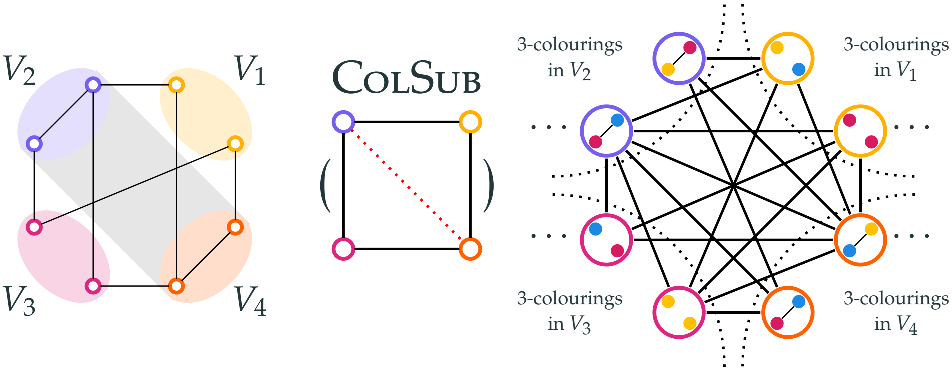
A framework of reduction: general patterns



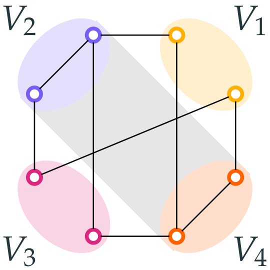
A framework of reduction: general patterns

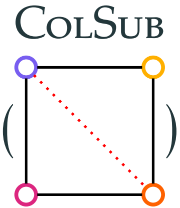


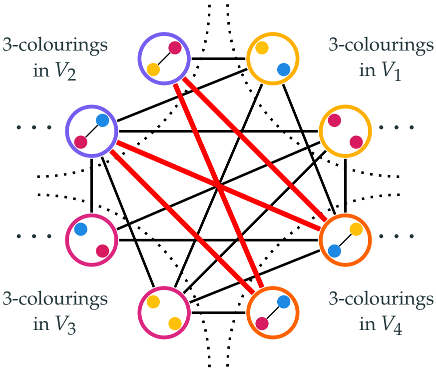
A framework of reduction: general patterns



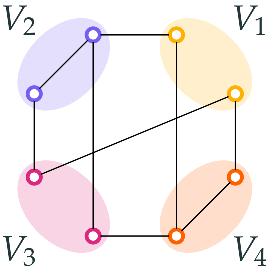
A framework of reduction: general patterns

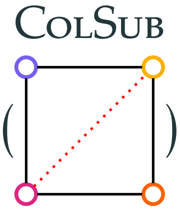


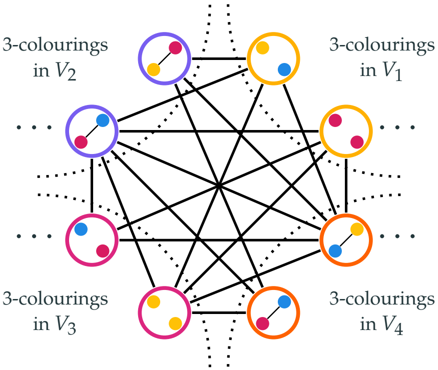




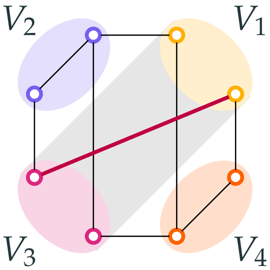
A framework of reduction: general patterns

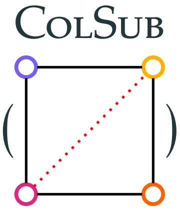


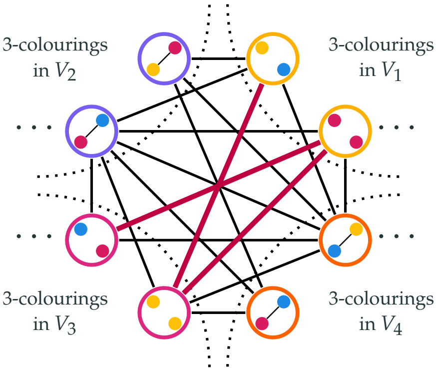


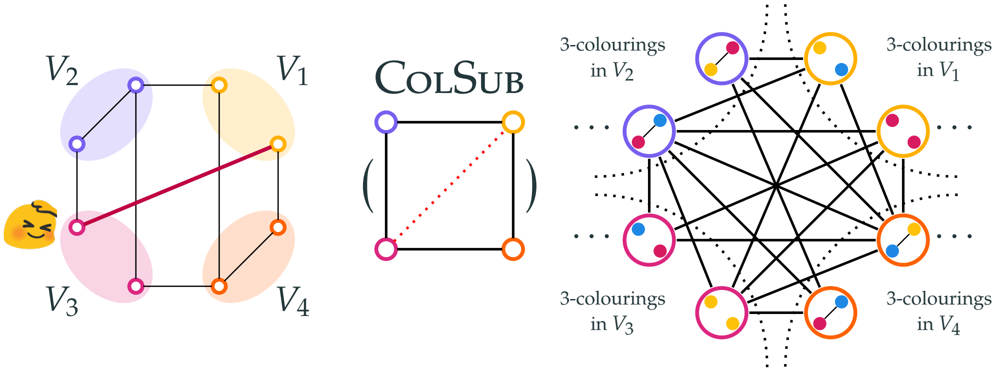


A framework of reduction: general patterns





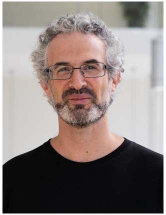




Core problem: Pacify the unhappy edges (Bad partitioning is in general inevitable)

Marx's approach

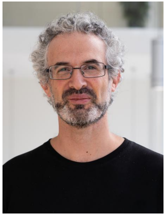
How do you prove this?



Marx's approach

How do you prove this?

I use **graph embeddings**.

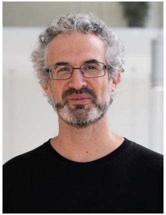


Marx's approach

How do you prove this?

I use **graph embeddings**.

Any proof strategy?



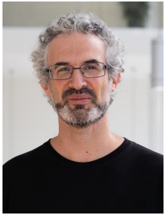
Marx's approach

How do you prove this?

I use **graph embeddings**.

Any proof strategy?

Large treewidth implies the existence of a large set without balance separators.



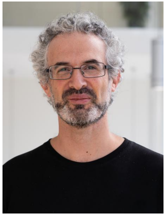
How do you prove this?

I use **graph embeddings**.

Any proof strategy?

Large treewidth implies the existence of a large set without balance separators.

A set without balanced separators implies the existence of a large uniform concurrent flow.



Marx's approach

How do you prove this?

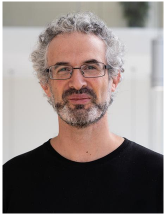
I use **graph embeddings**.

Any proof strategy?

Large treewidth implies the existence of a large set without balance separators.

A set without balanced separators implies the existence of a large uniform concurrent flow.

Paths in this concurrent flow can embed a blowup of the line graph of a complete graph.



Marx's approach

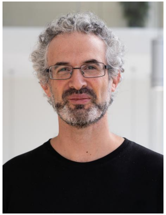
How do you prove this?

I use **graph embeddings**.

Any proof strategy?

Large treewidth implies the existence of a large set without balance separators.

A set without balanced separators implies the existence of a large uniform concurrent flow.



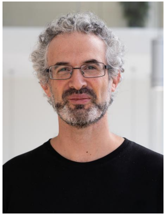
Paths in this concurrent flow can embed a blowup of the line graph of a complete graph.

Bounded-degree input graph can be embedded into this blowup of the complete graph.

Marx's approach

Uh... A bit too complicated. Any simpler proof?

And I only need the existence of hard patterns.

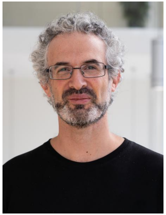


Marx's approach

Uh… A bit too complicated. Any simpler proof?

And I only need the existence of hard patterns.

Our SO**S**A'24 paper simplifies it!



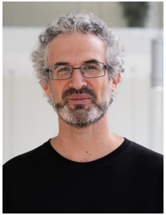
Marx's approach

Uh... A bit too complicated. Any simpler proof?

And I only need the existence of hard patterns.

Our SOSA'24 paper simplifies it!

Hard cases are expanders. (Their treewidth is large, too).



Marx's approach

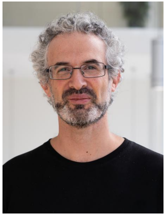
Uh... A bit too complicated. Any simpler proof?

And I only need the existence of hard patterns.

Our SOSA'24 paper simplifies it!

Hard cases are expanders. (Their treewidth is large, too).

Low-congested concurrent flow exists in expanders, due to **[Leighon–Rao'99]**!



Marx's approach

Uh... A bit too complicated. Any simpler proof?

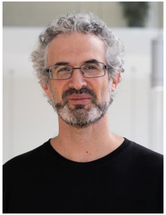
And I only need the existence of hard patterns.

Our SOSA'24 paper simplifies it!

Hard cases are expanders. (Their treewidth is large, too).

Low-congested concurrent flow exists in expanders, due to **[Leighon–Rao'99]**!

Then continue with the argument in the FOCS'07 paper.



Marx's approach

Uh... A bit too complicated. Any simpler proof?

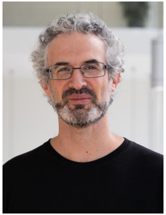
And I only need the existence of hard patterns.

Our SOSA'24 paper simplifies it!

Hard cases are expanders. (Their treewidth is large, too).

Low-congested concurrent flow exists in expanders, due to [Leighon–Rao'99]!

Then continue with the argument in the FOCS'07 paper.



Uh, okay...

Preliminaries of both papers

(Treewidth and brambles) + expanders + graph embedding + (Ramanujan graph / L_1-embeded LP rounding, choose one)

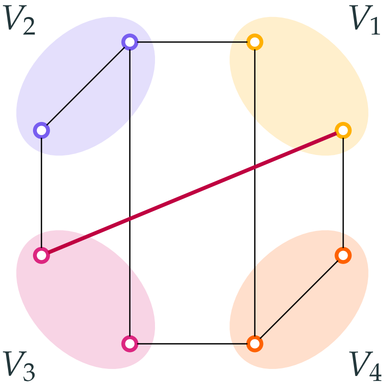
Preliminaries of both papers

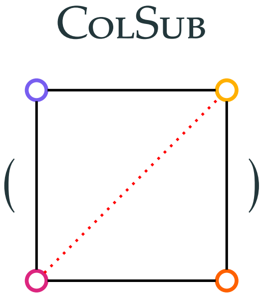
(Treewidth and brambles) + expanders + graph embedding + (Ramanujan graph / L_1-embeded LP rounding, choose one)

Goal

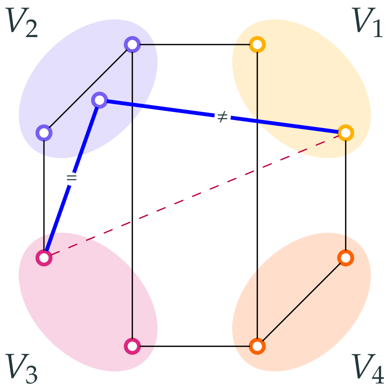
Can we give a **white-boxed proof** that even **second-year undergraduate students** can understand?!

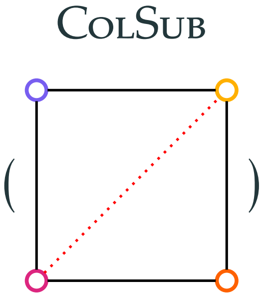
Our new perspective



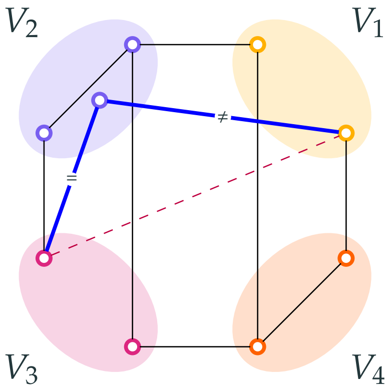


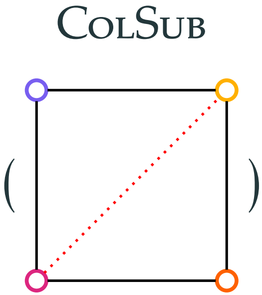
Our new perspective





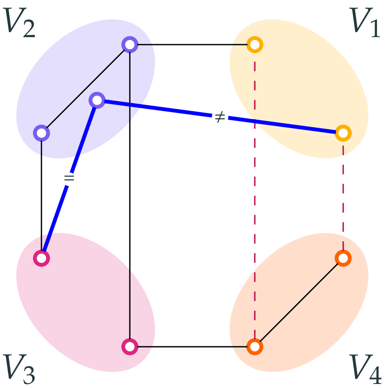
Our new perspective





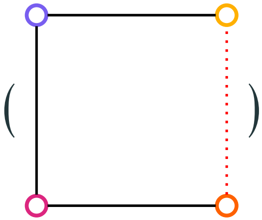
But this costs us something...

Our new perspective

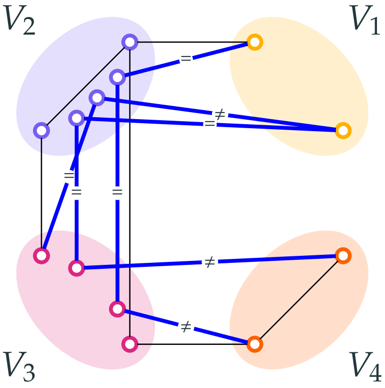


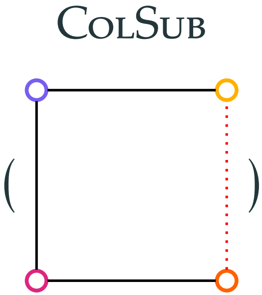
But this costs us something...

ColSub

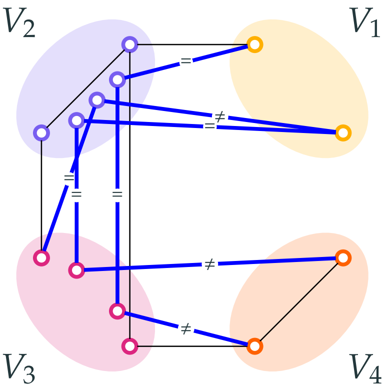


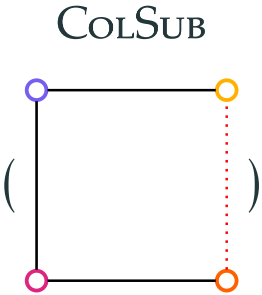
Our new perspective





But this costs us something...





But this costs us something... Too many new vertices in V_2!

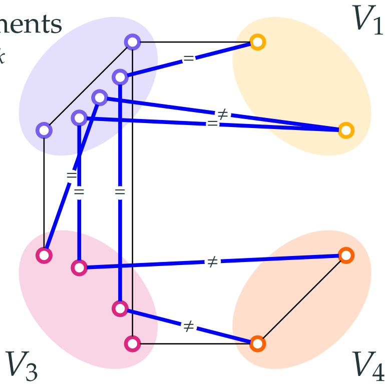
Our new perspective

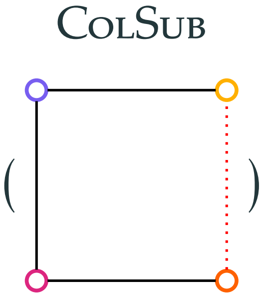
#vertices

$\gg n/k$

#3-assignments

$\gg 3^{n/k}$





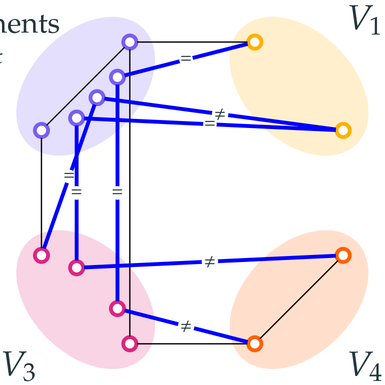
But this costs us something... Too many new vertices in V_2!

#vertices

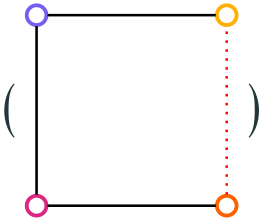
$\gg n/k$

#3-assignments

$\gg 3^{n/k}$



COLSUB



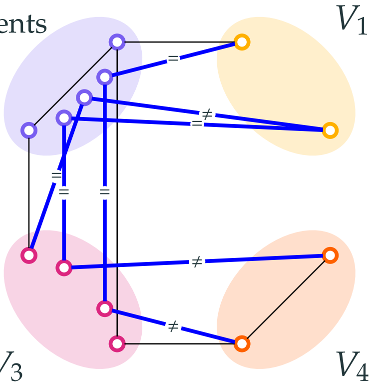
But this costs us something... Too many new vertices in V_2!

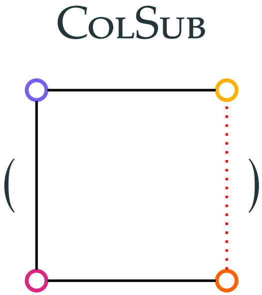
Routing in paths are highly **congested**!

Our new perspective

#vertices $\gg n/k$

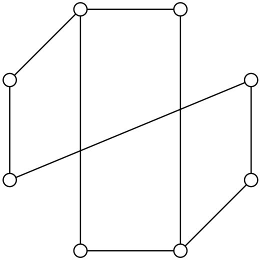
#3-assignments $\gg 3^{n/k}$



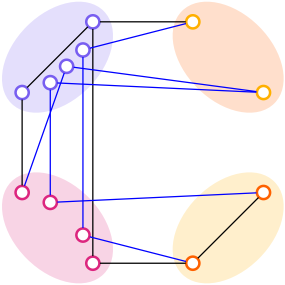


But this costs us something... Too many new vertices in V_2!

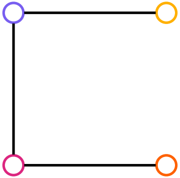
Routing in paths are highly **congested**! Indeed, ColSub(path) is FPT.



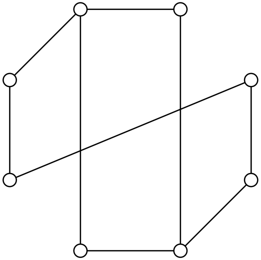
G



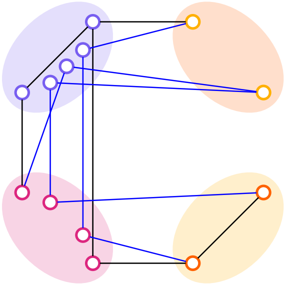
G'



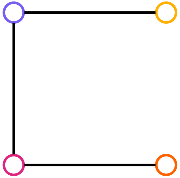
H



G



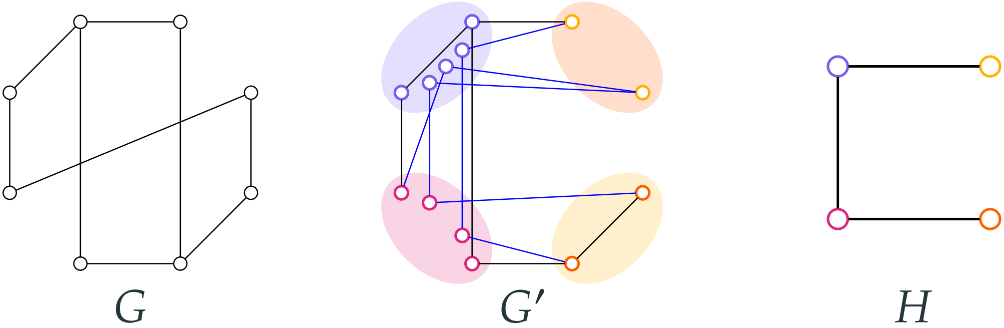
G'



H

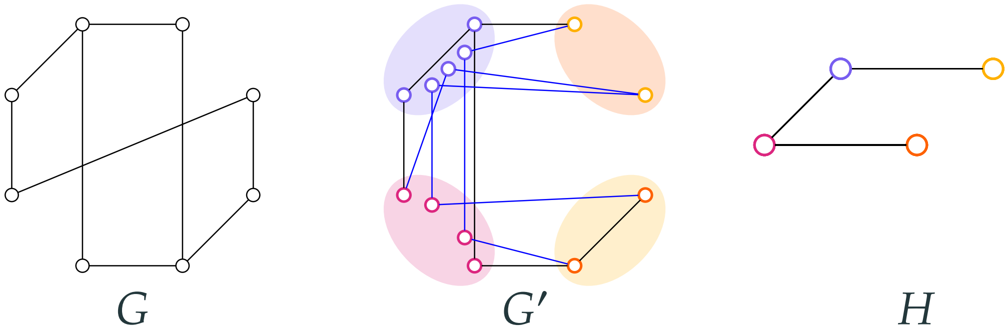
Topological minor

G is a **topological minor** of G' if G' is (isomorphic to) an edge-subdivision of G.



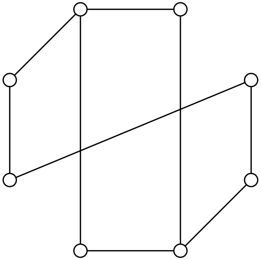
Topological minor

G is a **topological minor** of G' if G' is (isomorphic to) an edge-subdivision of G.

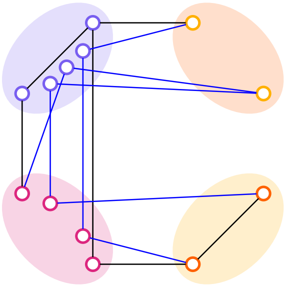


Topological minor

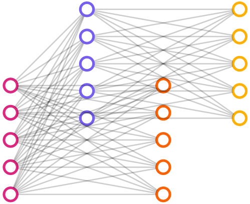
G is a **topological minor** of G' if G' is (isomorphic to) an edge-subdivision of G.



G



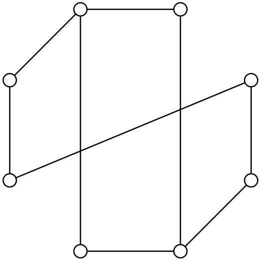
G'



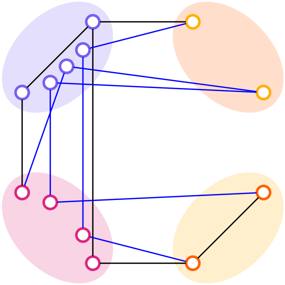
$H \otimes J_5$

Topological minor

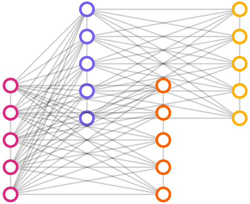
G is a **topological minor** of G' if G' is (isomorphic to) an edge-subdivision of G.



G



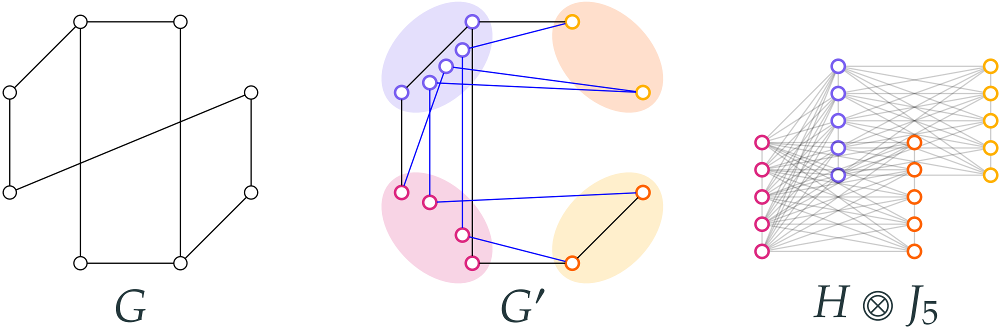
G'



$H \otimes J_5$

Blowup

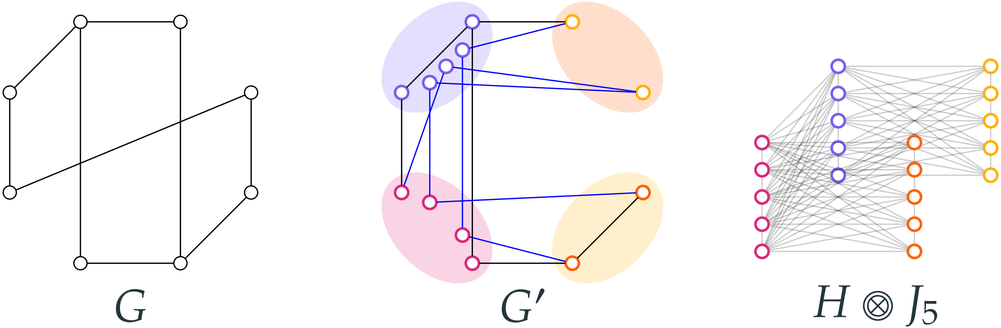
A t-**blowup graph** is obtained by replacing vertices with K_t and edges with $K_{t,t}$.



Goal: Any degree-4 graph G is a topological minor of $H \otimes J_{t'}$.

Ideally, t' should be as small as possible. (Though it must be $\geq n/k$.)

Fitting instance G into Blowups $H \otimes J_t$

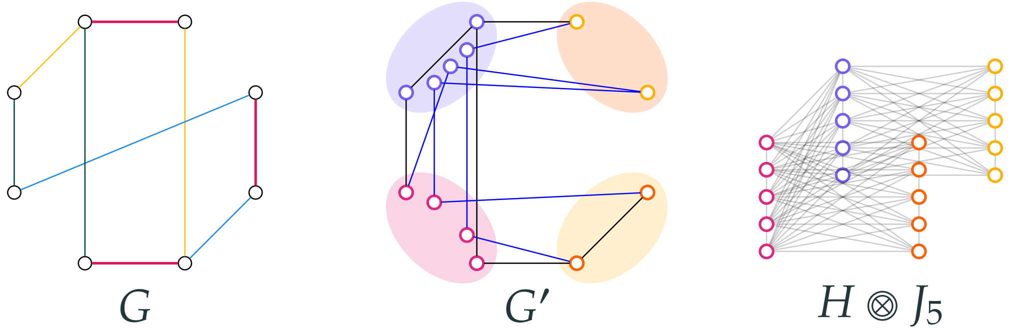


Goal: Any degree-4 graph G is a topological minor of $H \otimes J_{t'}$.
Ideally, t' should be as small as possible. (Though it must be $\geq n/k$.)

Let $\gamma'(H)$ be the maximum such that the minimum of t' is $n/\gamma'(H)$.

ETH implies an $n^{\Omega(\gamma'(H))}$ lower bound on $\textsc{ColSub}(H)$.

Fitting instance G into Blowups $H \otimes J_t$



Goal: Any degree-4 graph G is a topological minor of $H \otimes J_{t'}$.
Ideally, t' should be as small as possible. (Though it must be $\geq n/k$.)

New goal: Any **matching** on n vertices is a topological minor of $H \otimes J_t$.

Vizing's Theorem + definition of blowup: $t' \leq 5t$.

Linkage

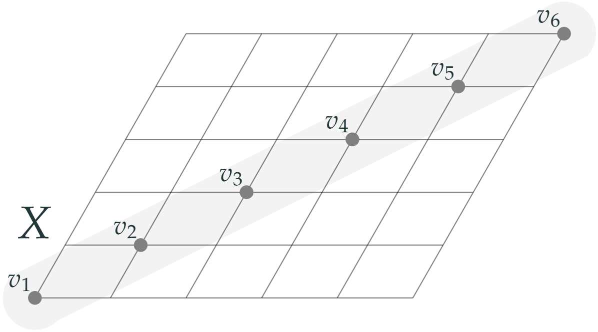
H is a small graph (k vertices), but after blowing up, it needs to 'host' a large matching (n vertices)...

Observation: If there are γ vertices in H that can 'host' **any** matching among them (via routing through other vertices), then a $t = n/\gamma$ blowup of H suffices!

Linkage

H is a small graph (k vertices), but after blowing up, it needs to 'host' a large matching (n vertices)...

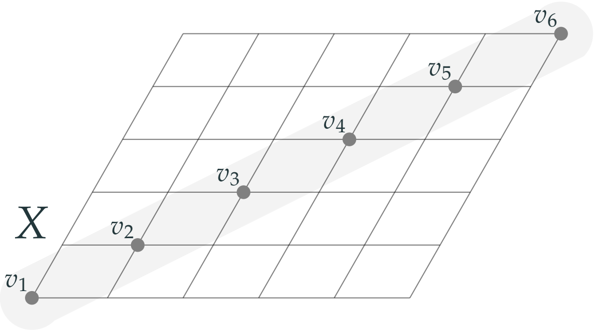
Observation: If there are γ vertices in H that can 'host' **any** matching among them (via routing through other vertices), then a $t = n/\gamma$ blowup of H suffices!



Linkage

H is a small graph (k vertices), but after blowing up, it needs to 'host' a large matching (n vertices)...

Observation: If there are γ vertices in H that can 'host' **any** matching among them (via routing through other vertices), then a $t = n/\gamma$ blowup of H suffices!

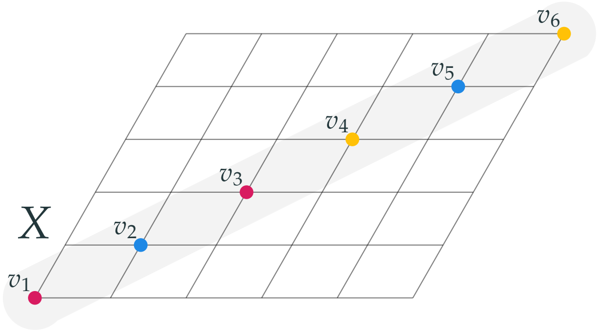


M-linkage

Linkage

H is a small graph (k vertices), but after blowing up, it needs to 'host' a large matching (n vertices)...

Observation: If there are γ vertices in H that can 'host' **any** matching among them (via routing through other vertices), then a $t = n/\gamma$ blowup of H suffices!

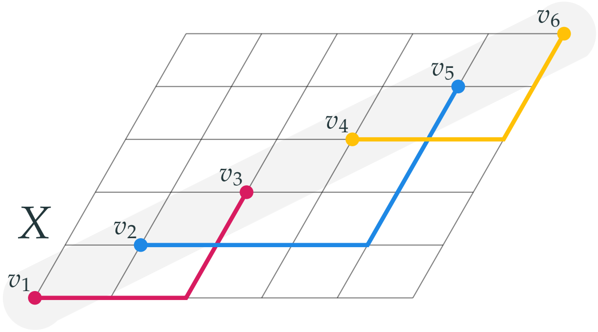


M-linkage

$$M = \{v_1v_3, v_2v_5, v_4v_6\}$$

H is a small graph (k vertices), but after blowing up, it needs to 'host' a large matching (n vertices)...

Observation: If there are γ vertices in H that can 'host' **any** matching among them (via routing through other vertices), then a $t = n/\gamma$ blowup of H suffices!



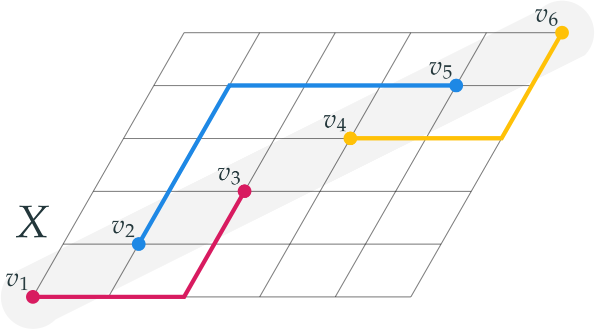
M-linkage

$$M = \{v_1v_3, v_2v_5, v_4v_6\}$$

No vertex can be used twice

H is a small graph (k vertices), but after blowing up, it needs to 'host' a large matching (n vertices)...

Observation: If there are γ vertices in H that can 'host' **any** matching among them (via routing through other vertices), then a $t = n/\gamma$ blowup of H suffices!



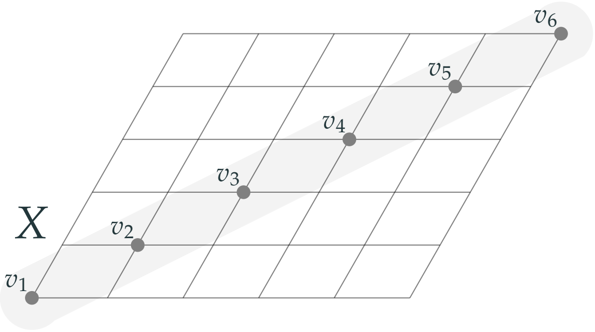
M-linkage

$$M = \{v_1v_3, v_2v_5, v_4v_6\}$$

Congestion-free communication

H is a small graph (k vertices), but after blowing up, it needs to 'host' a large matching (n vertices)...

Observation: If there are γ vertices in H that can 'host' **any** matching among them (via routing through other vertices), then a $t = n/\gamma$ blowup of H suffices!

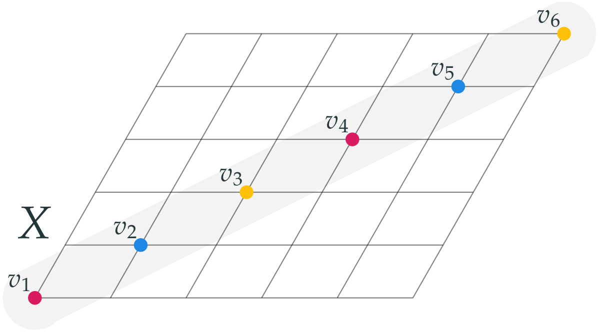


Matching-linked

X is matching-linked, iff there is an M-linkage for all matchings M over X.

H is a small graph (k vertices), but after blowing up, it needs to 'host' a large matching (n vertices)...

Observation: If there are γ vertices in H that can 'host' **any** matching among them (via routing through other vertices), then a $t = n/\gamma$ blowup of H suffices!



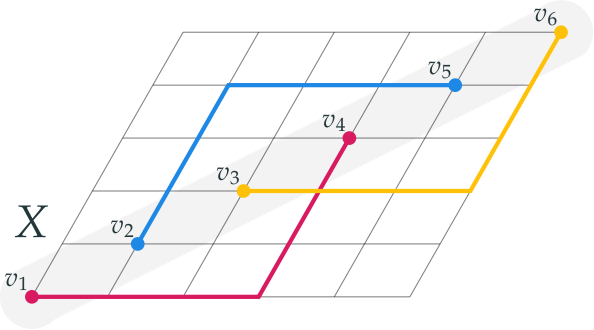
Matching-linked

X is matching-linked, iff there is an M-linkage for all matchings M over X.

$$M = \{v_1v_4, v_2v_5, v_3v_6\}$$

H is a small graph (k vertices), but after blowing up, it needs to 'host' a large matching (n vertices)...

Observation: If there are γ vertices in H that can 'host' **any** matching among them (via routing through other vertices), then a $t = n/\gamma$ blowup of H suffices!



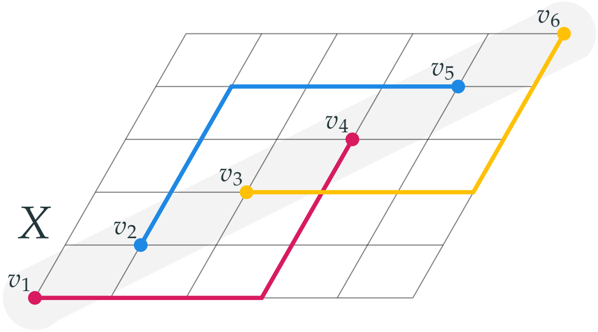
Matching-linked

X is matching-linked, iff there is an M-linkage for all matchings M over X.

$$M = \{v_1v_4, v_2v_5, v_3v_6\}$$

H is a small graph (k vertices), but after blowing up, it needs to 'host' a large matching (n vertices)...

Observation: If there are γ vertices in H that can 'host' **any** matching among them (via routing through other vertices), then a $t = n/\gamma$ blowup of H suffices!



Matching-linked

X is matching-linked, iff there is an M-linkage for all matchings M over X.

$$M = \{v_1v_4, v_2v_5, v_3v_6\}$$

Not matching-linked!

Theorem

[Marx'10]

There is a sequence of **degree-4** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and $\textsc{ColSub}(H_\ell)$ cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.

It suffices to find a graph H that

Theorem

[Marx'10]

There is a sequence of **degree-4** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and $\textsc{ColSub}(H_\ell)$ cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.

It suffices to find a graph H that

(1) has $O(s \log s)$ vertices, (2) is of max degree 4,

and (3) has a matching-linked set of size s.

Theorem

[Marx'10]

There is a sequence of **degree-4** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and $\textsc{ColSub}(H_\ell)$ cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.

It suffices to find a graph H that

(1) has $O(s \log s)$ vertices, (2) is of max degree 4,
and (3) has a matching-linked set of size s.

Our solution: Beneš network

coined by Václav Beneš in Bell lab in 1964

Theorem **[Marx'10]**

There is a sequence of **degree-4** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and $\textsc{ColSub}(H_\ell)$ cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.

It suffices to find a graph H that

(1) has $O(s \log s)$ vertices, (2) is of max degree 4,
and (3) has a matching-linked set of size s.

Our solution: Beneš network

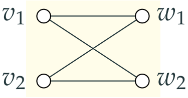
coined by Václav Beneš in Bell lab in 1964

Fun fact: it is NOT an expander.

The Beneš network

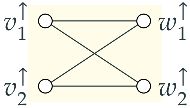
... is two butterflies back to back.

$B_1 =$

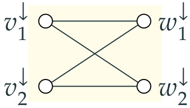


The Beneš network

$B_1^{\uparrow} =$



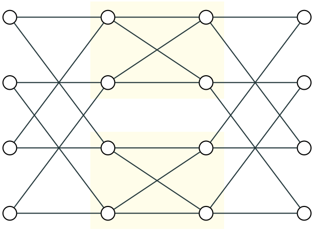
$B_1^{\downarrow} =$

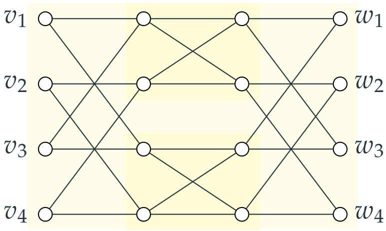


The Beneš network

$$B_1^{\uparrow} =$$

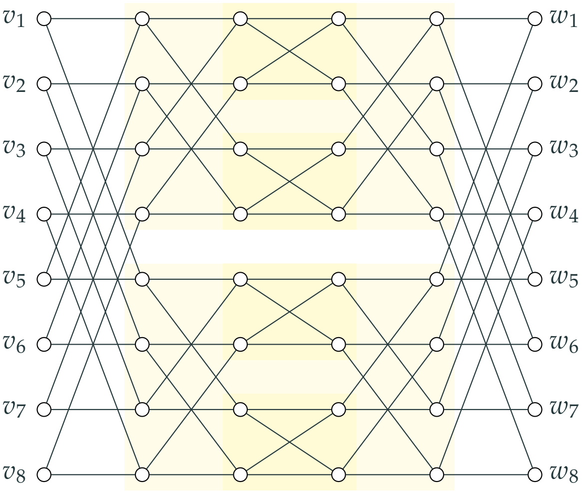
$$B_1^{\downarrow} =$$



$B_2 =$ 

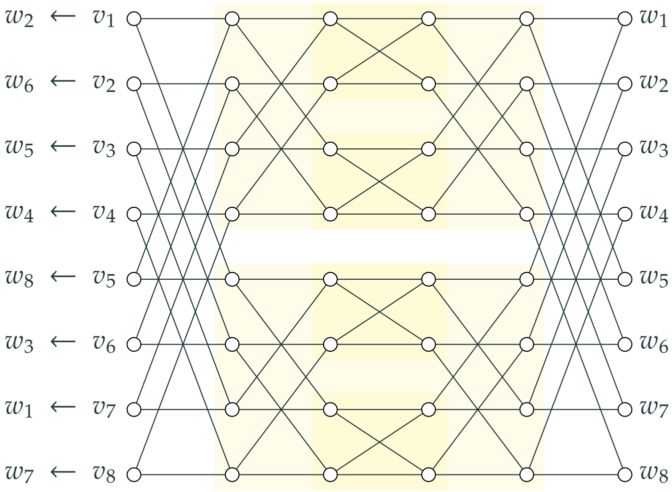
The Beneš network

$B_3 =$



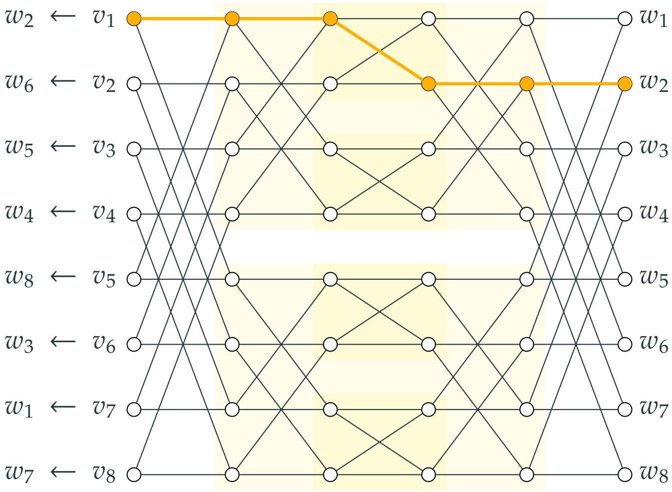
Theorem [Beneš'64]

For any permutation π over $[2^s]$, there is a $\{v_i w_{\pi(i)}\}$-linkage in B_s.



Theorem **[Beneš'64]**

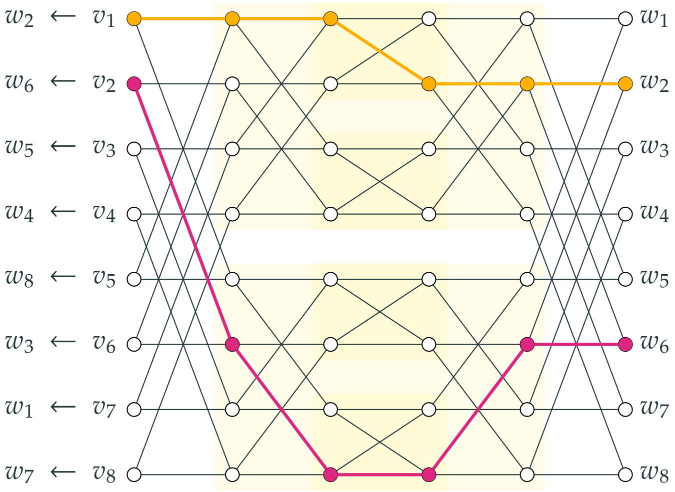
For any permutation π over $[2^s]$, there is a $\{v_i w_{\pi(i)}\}$-linkage in B_s.



The Beneš network

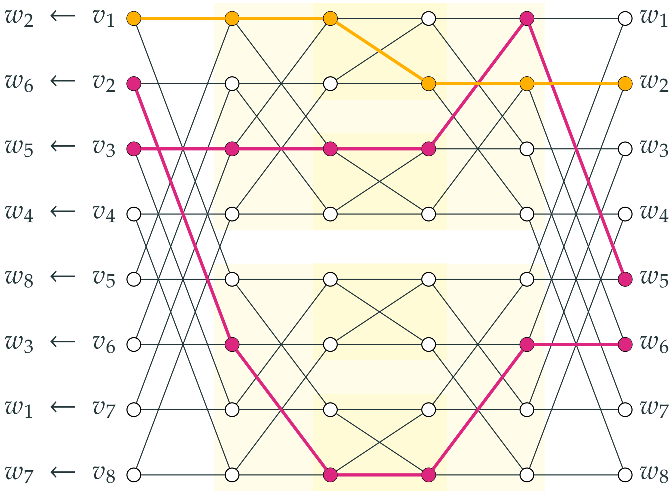
Theorem [Beneš'64]

For any permutation π over $[2^s]$, there is a $\{v_i w_{\pi(i)}\}$-linkage in B_s.



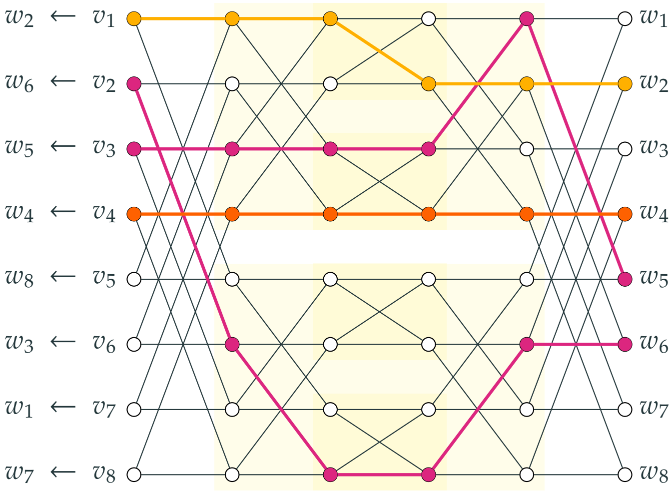
Theorem [Beneš'64]

For any permutation π over $[2^s]$, there is a $\{v_i w_{\pi(i)}\}$-linkage in B_s.



Theorem [Beneš'64]

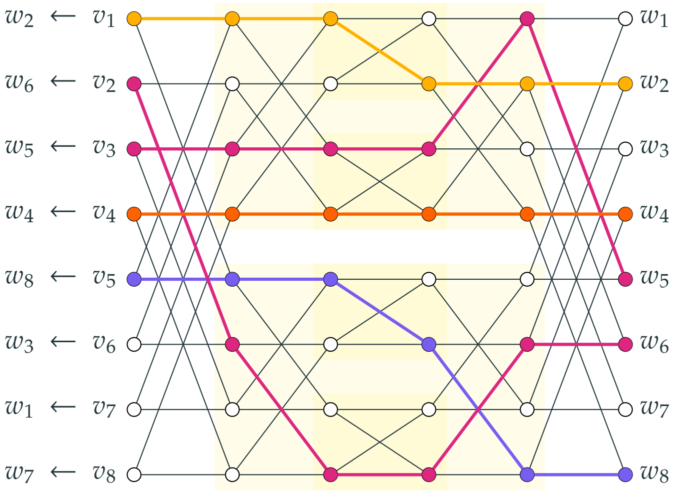
For any permutation π over $[2^s]$, there is a $\{v_i w_{\pi(i)}\}$-linkage in B_s.



Theorem

[Beneš'64]

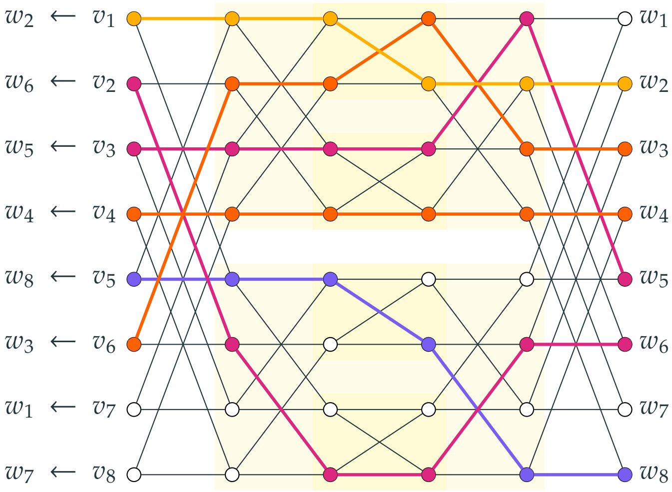
For any permutation π over $[2^s]$, there is a $\{v_i w_{\pi(i)}\}$-linkage in B_s.



The Beneš network

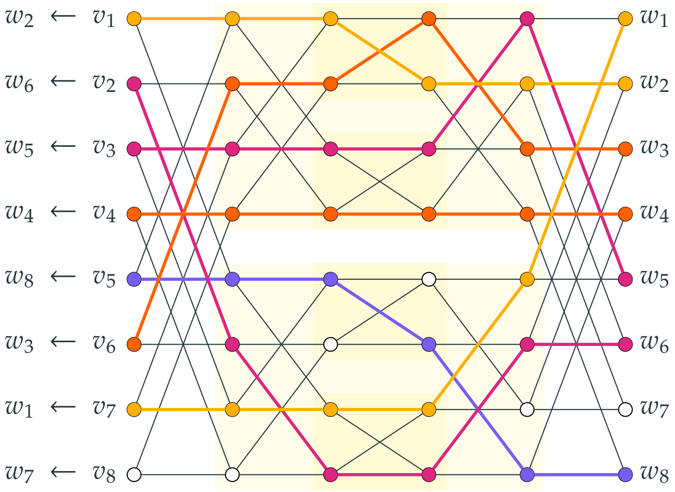
Theorem **[Beneš'64]**

For any permutation π over $[2^s]$, there is a $\{v_i w_{\pi(i)}\}$-linkage in B_s.



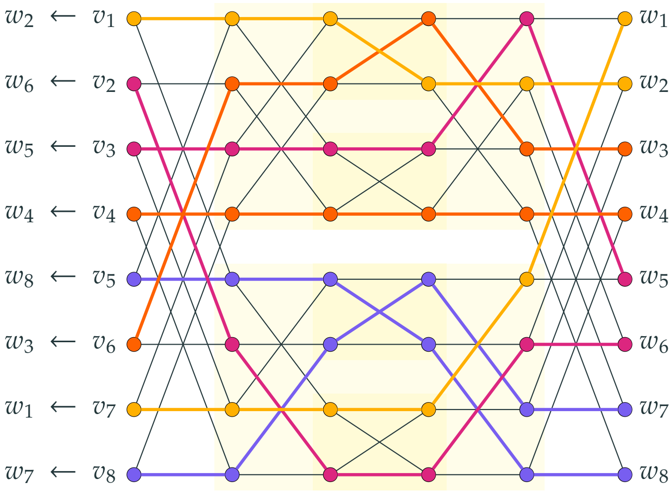
Theorem [Beneš'64]

For any permutation π over $[2^s]$, there is a $\{v_i w_{\pi(i)}\}$-linkage in B_s.

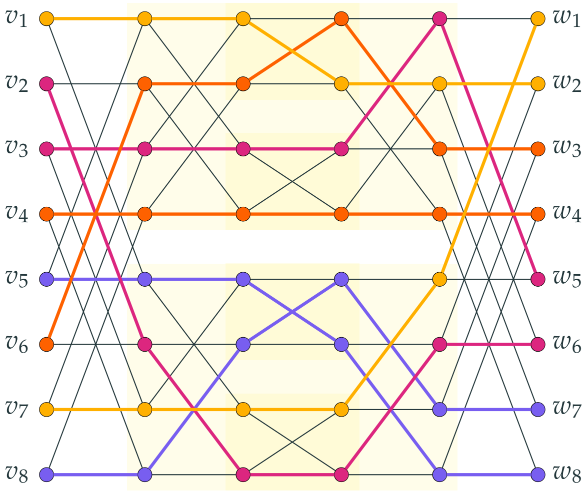


Theorem [Beneš'64]

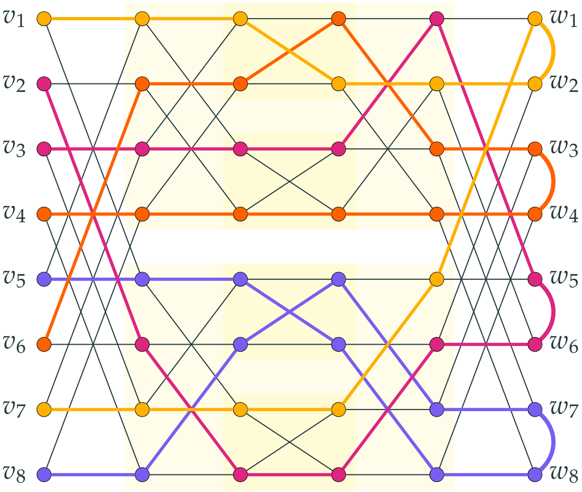
For any permutation π over $[2^s]$, there is a $\{v_i w_{\pi(i)}\}$-linkage in B_s.



M-linkage for $M = \{v_1v_7,\ v_2v_3,\ v_4v_6,\ v_5v_8\}$?

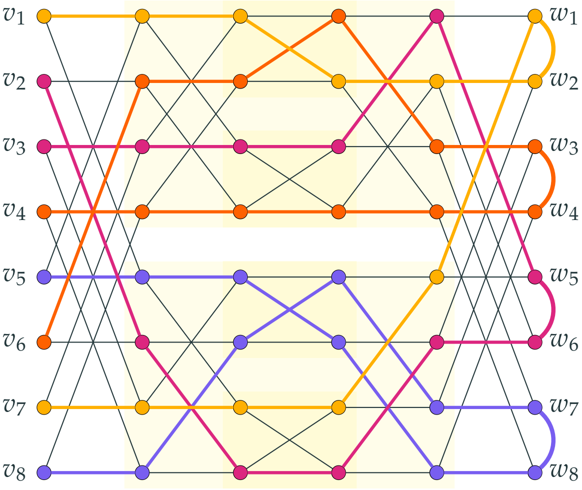


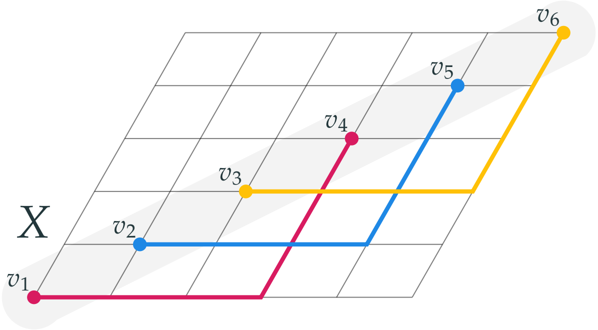
M-linkage for $M = \{v_1v_7,\ v_2v_3,\ v_4v_6,\ v_5v_8\}$?



The augmented Beneš network

$\{v_1, \cdots, v_{2^s}\}$ is matching-linked in $\check{B}_s$!



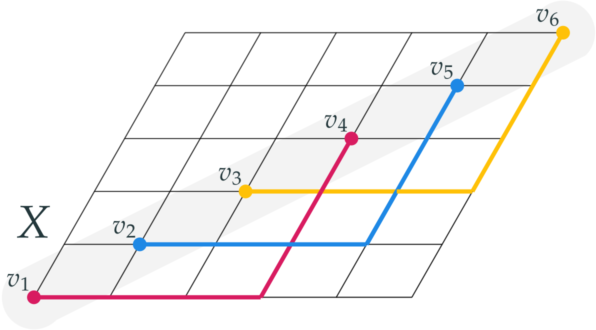


Matching-linked

X is matching-linked, iff there is an M-linkage for all matchings M over X.

$$M = \{v_1v_4, v_2v_5, v_3v_6\}$$

Not matching-linked!

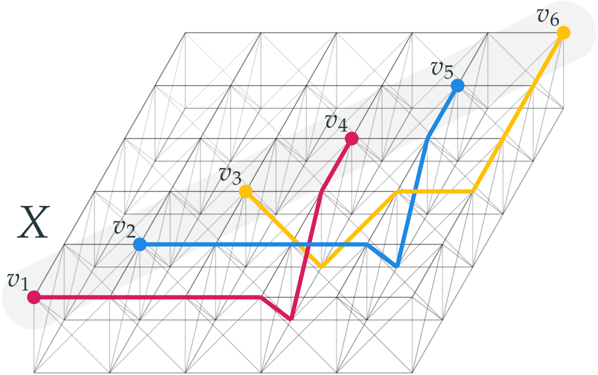


Matching-linked

X is matching-linked, iff there is an M-linkage for all matchings M over X.

$$M = \{v_1v_4, v_2v_5, v_3v_6\}$$

Not matching-linked!



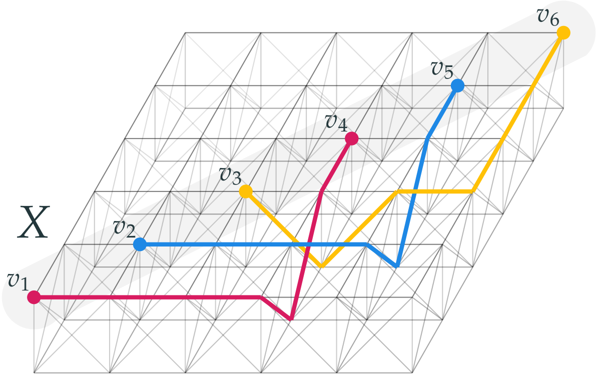
Matching-linked

X is matching-linked, iff there is an M-linkage for all matchings M over X.

$$M = \{v_1v_4, v_2v_5, v_3v_6\}$$

Not matching-linked!

Matching-linked in 2-blowup.



Matching-linked

X is matching-linked, iff there is an M-linkage for all matchings M over X.

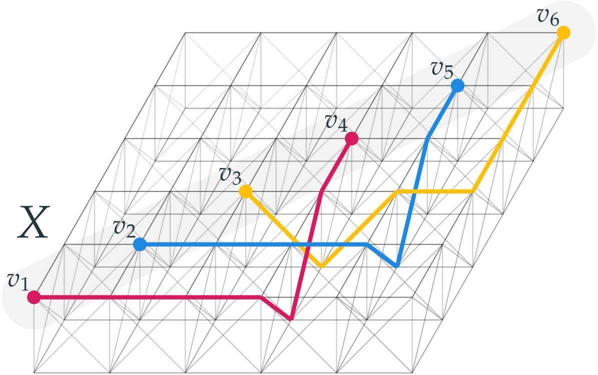
$$M = \{v_1 v_4, v_2 v_5, v_3 v_6\}$$

Not matching-linked!

Matching-linked in 2-blowup.

Linkage capacity

The linkage capacity $\gamma(H)$ is the supermum over $c > 0$ such that $H \otimes J_t$ contains a matching-linked set with $|X| = \lfloor ct \rfloor$ for all large t.



Matching-linked

X is matching-linked, iff there is an M-linkage for all matchings M over X.

$$M = \{v_1v_4, v_2v_5, v_3v_6\}$$

Not matching-linked!

Matching-linked in 2-blowup.

Theorem (main, rough statement)

For **any** graph H, $\textsc{ColSub}(H)$ cannot be solved in time $n^{o(\gamma(H))}$ unless ETH fails.

Hardness based on linkage capacity

Unless ETH fails, ColSub(H), for a k-vertex graph H, cannot be solved in time

Hardness based on linkage capacity

Unless ETH fails, ColSub(H), for a k-vertex graph H, cannot be solved in time

- $n^{o(d)}$, for **any** graph H with **average degree** d;

Hardness based on linkage capacity

Unless ETH fails, ColSub(H), for a k-vertex graph H, cannot be solved in time

- $n^{o(d)}$, for **any** graph H with **average degree** d;
  - Asymptotically optimal.

Hardness based on linkage capacity

Unless ETH fails, ColSub(H), for a k-vertex graph H, cannot be solved in time

- $n^{o(d)}$, for **any** graph H with **average degree** d;
  - Asymptotically optimal.
- $n^{o(k)}$, for **almost every** k-vertex graph H with **polynomial average degree**;

Hardness based on linkage capacity

Unless ETH fails, ColSub(H), for a k-vertex graph H, cannot be solved in time

- $n^{o(d)}$, for **any** graph H with **average degree** d;
  - Asymptotically optimal.
- $n^{o(k)}$, for **almost every** k-vertex graph H with **polynomial average degree**;
  - Asymptotically optimal.

Hardness based on linkage capacity

Unless ETH fails, ColSub(H), for a k-vertex graph H, cannot be solved in time

- $n^{o(d)}$, for **any** graph H with **average degree** d;
  - Asymptotically optimal.
- $n^{o(k)}$, for **almost every** k-vertex graph H with **polynomial average degree**;
  - Asymptotically optimal.
- $n^{o(t)}$, for **any minor-free** graph (family) with **treewidth** $t = \mathrm{tw}(H)$;

Hardness based on linkage capacity

Unless ETH fails, ColSub(H), for a k-vertex graph H, cannot be solved in time

- $n^{o(d)}$, for **any** graph H with **average degree** d;
  - Asymptotically optimal.
- $n^{o(k)}$, for **almost every** k-vertex graph H with **polynomial average degree**;
  - Asymptotically optimal.
- $n^{o(t)}$, for **any minor-free** graph (family) with **treewidth** $t = \mathrm{tw}(H)$;
  - Asymptotically optimal.

Hardness based on linkage capacity

Unless ETH fails, ColSub(H), for a k-vertex graph H, cannot be solved in time

- $n^{o(d)}$, for **any** graph H with **average degree** d;
  - Asymptotically optimal.
- $n^{o(k)}$, for **almost every** k-vertex graph H with **polynomial average degree**;
  - Asymptotically optimal.
- $n^{o(t)}$, for **any minor-free** graph (family) with **treewidth** $t = \mathrm{tw}(H)$;
  - Asymptotically optimal.
- $n^{o(t/\log t)}$, for **any** graph with **treewidth** $t = \mathrm{tw}(H)$.

Hardness based on linkage capacity

Unless ETH fails, $\textsc{ColSub}(H)$, for a k-vertex graph H, cannot be solved in time

- $n^{o(d)}$, for **any** graph H with **average degree** d;
  - Asymptotically optimal.
- $n^{o(k)}$, for **almost every** k-vertex graph H with **polynomial average degree**;
  - Asymptotically optimal.
- $n^{o(t)}$, for **any minor-free** graph (family) with **treewidth** $t = \mathrm{tw}(H)$;
  - Asymptotically optimal.
- $n^{o(t/\log t)}$, for **any** graph with **treewidth** $t = \mathrm{tw}(H)$.
  - New proof to Marx's "Can you beat treewidth?" theorem.

Hardness based on linkage capacity

Unless ETH fails, ColSub(H), for a k-vertex graph H, cannot be solved in time

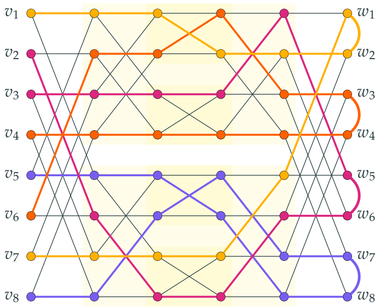
- $n^{o(d)}$, for **any** graph H with **average degree** d;
  - Asymptotically optimal.
- $n^{o(k)}$, for **almost every** k-vertex graph H with **polynomial average degree**;
  - Asymptotically optimal.
- $n^{o(t)}$, for **any minor-free** graph (family) with **treewidth** $t = \mathrm{tw}(H)$;
  - Asymptotically optimal.
- $n^{o(t/\log t)}$, for **any** graph with **treewidth** $t = \mathrm{tw}(H)$.
  - New proof to Marx's "Can you beat treewidth?" theorem.

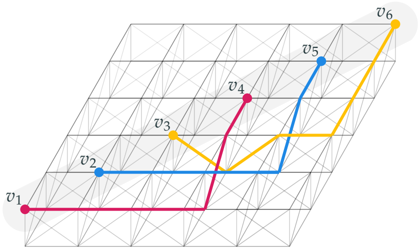
Implications to *induced subgraph counting*.

[Roth–Schmitt–Wellnitz'20, Döring–Marx–Wellnitz'24a,24b, Curticapean–Neuen'24]

Summary

Hardness of subgraph counting via **linkage**.

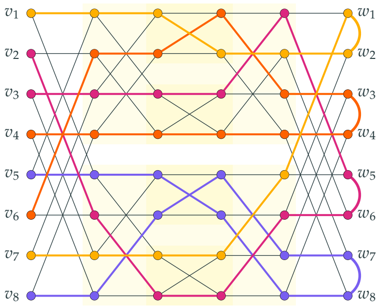


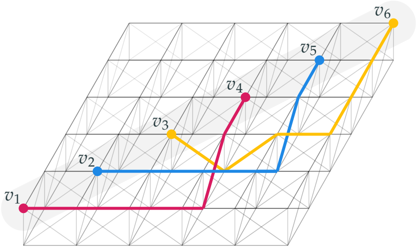


Summary

Hardness of subgraph counting via **linkage**.

Beneš network for $n^{\Omega(k/\log k)}$ lower bound.



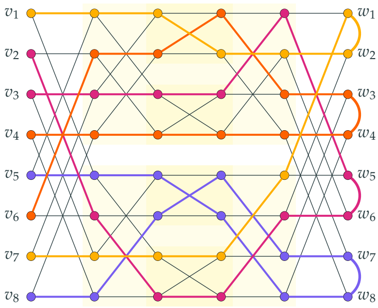


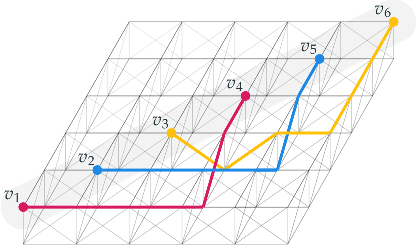
Summary

Hardness of subgraph counting via **linkage**.

Beneš network for $n^{\Omega(k/\log k)}$ lower bound.

Hardness of general patterns via **linkage capacity**.



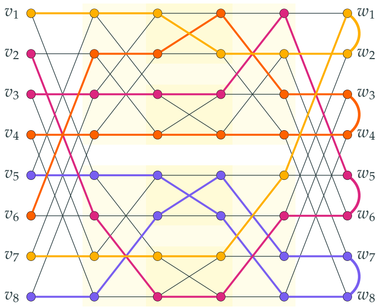


Summary

Hardness of subgraph counting via **linkage**.

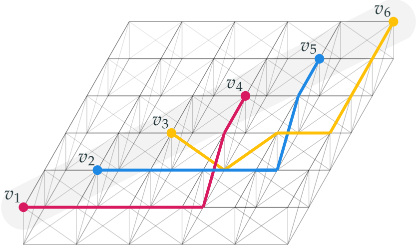
Beneš network for $n^{\Omega(k/\log k)}$ lower bound.

Hardness of general patterns via **linkage capacity**.



Open questions A:

Close the gap between $n^{\Omega(k/\log k)}$ lower bound and $n^{O(k)}$ algorithms?

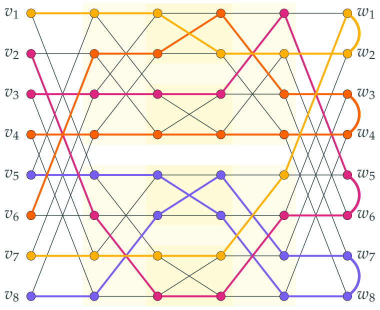


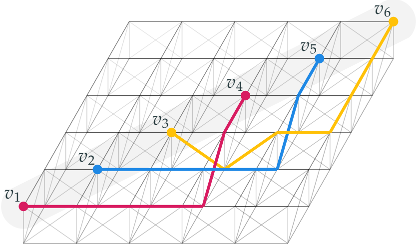
Summary

Hardness of subgraph counting via **linkage**.

Beneš network for $n^{\Omega(k/\log k)}$ lower bound.

Hardness of general patterns via **linkage capacity**.





Open questions A:

Close the gap between $n^{\Omega(k/\log k)}$ lower bound and $n^{O(k)}$ algorithms?

Can you beat treewidth? ($n^{\Omega(\mathrm{tw}(H))}$ lower bound?)

Summary

Hardness of subgraph counting via **linkage**.

Beneš network for $n^{\Omega(k/\log k)}$ lower bound.

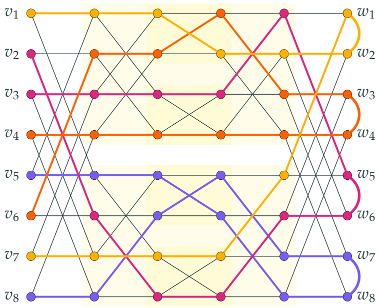
Hardness of general patterns via **linkage capacity**.

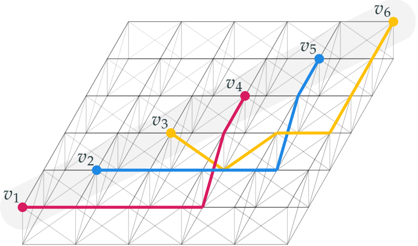
Open questions A:

Close the gap between $n^{\Omega(k/\log k)}$ lower bound and $n^{O(k)}$ algorithms?

Can you beat treewidth? ($n^{\Omega(\mathrm{tw}(H))}$ lower bound?)

Design algorithms based on linkage capacity? ($n^{O(\gamma(H))}$ algorithm?)

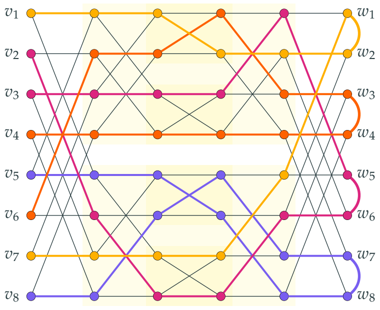




Hardness of subgraph counting via **linkage**.

Beneš network for $n^{\Omega(k/\log k)}$ lower bound.

Hardness of general patterns via **linkage capacity**.



Open questions B:

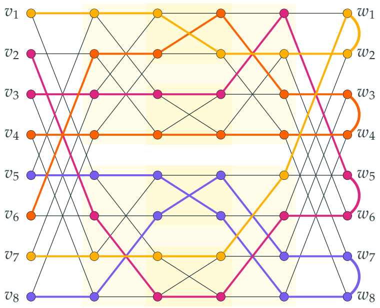
Novel usages of communication networks?

- This has been used in extension complexity **[Göös–Jain–Watson'18]**, and "fine-grained" hardness of gap amplification (PCP) **[Bafna–Minzer–Vyas'24]**.

Hardness of subgraph counting via **linkage**.

Beneš network for $n^{\Omega(k/\log k)}$ lower bound.

Hardness of general patterns via **linkage capacity**.



Open questions B:

Novel usages of communication networks?

- This has been used in extension complexity **[Göös–Jain–Watson'18]**, and "fine-grained" hardness of gap amplification (PCP) **[Bafna–Minzer–Vyas'24]**.

New proofs of $t/\log t$ lower bounds in other settings that use treewidth-separator duality?

- E.g., $\mathbf{AC}^0$ lower bounds for subgraph isomorphism **[Li–Razborov–Rossman'17]**?

Thank you!

arXiv: 2410.02606