Warning

These slides contain **animations** that might not work properly in some PDF reader.

We kindly ask you to open this document in a supported PDF reader.

Known to work: Adobe Acrobat

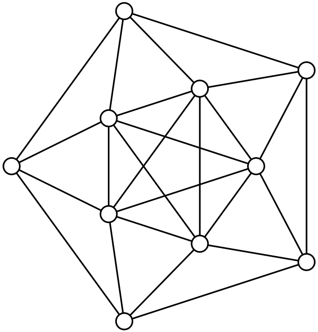
Known to fail: any web browser, SumatraPDF

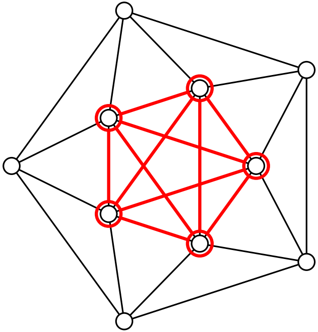
M1 University of Regensburg
M2 ITU Copenhagen
M3 Max-Planck Institute
M4 Saarland University

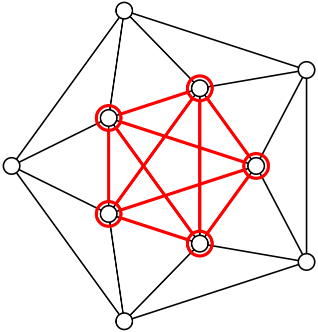
Can You Link Up With Treewidth?

Radu Curticapean M1 M2
Simon Döring M3 M4
Daniel Neuen M3
Jiaheng Wang M1

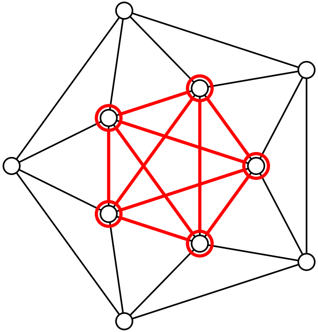
2nd July 2025, Paris





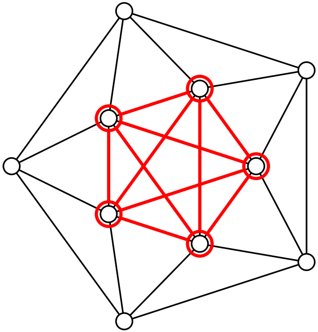


n^k	trivial algorithm



~~poly(n, k)~~ **NP**-hardness

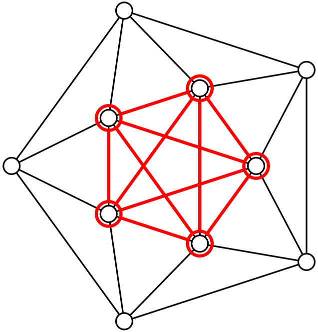
n^k trivial algorithm



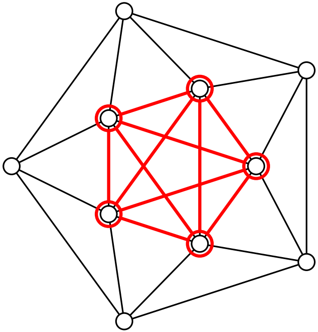
~~$\text{poly}(n, k)$~~ **NP**-hardness

$f(k) \cdot \text{poly}(n)$? **FPT** time?

n^k trivial algorithm



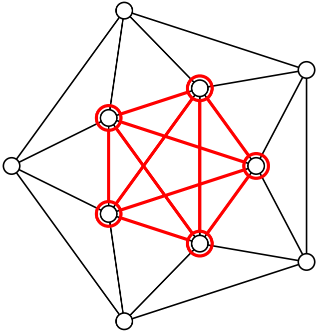
~~poly(n, k)~~	**NP**-hardness
$k^k \cdot O(n^{42})$?	**FPT** time?
n^k	trivial algorithm



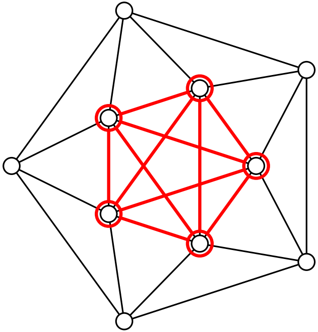
~~$\text{poly}(n, k)$~~ **NP**-hardness

~~$f(k) \cdot \text{poly}(n)$~~ **W[1]**-hardness

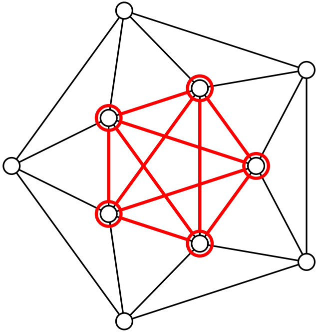
n^k trivial algorithm



~~$\text{poly}(n, k)$~~	**NP**-hardness
~~$f(k) \cdot \text{poly}(n)$~~	**W[1]**-hardness
$f(k) \cdot n^{o(k)}$?	
n^k	trivial algorithm

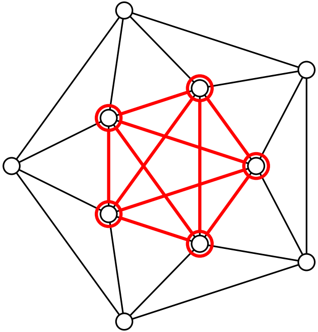


~~$\text{poly}(n,k)$~~	**NP**-hardness
~~$f(k)\cdot\text{poly}(n)$~~	**W[1]**-hardness
$n^{O(\sqrt{k})}$?	
n^k	trivial algorithm



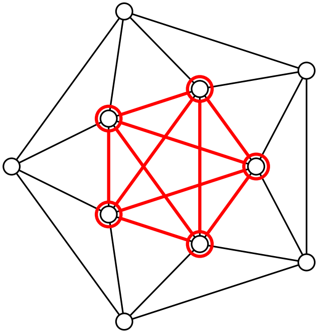
~~$\text{poly}(n,k)$~~	**NP**-hardness
~~$f(k)\cdot\text{poly}(n)$~~	**W[1]**-hardness
~~$f(k)\cdot n^{o(k)}$~~	Exponential-time hypothesis
n^k	trivial algorithm

Is CLIQUE a good source of hardness?



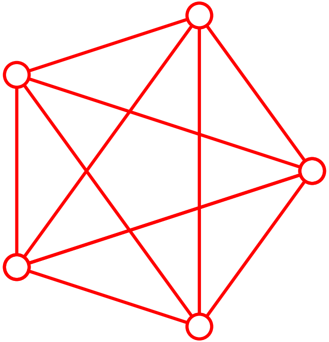
no $n^{o(k)}$ algorithm

Is Clique a good source of hardness?



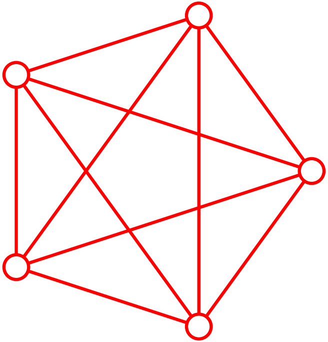
no $n^{o(k)}$ algorithm

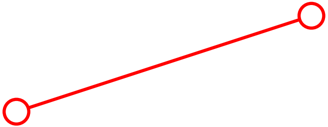
Is Clique a good source of hardness?



no $n^{o(k)}$ algorithm

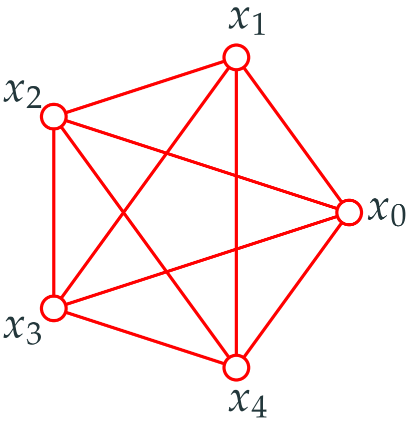
Is Clique a good source of hardness?



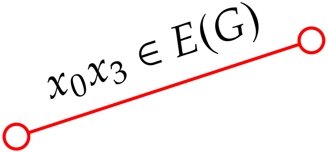


no $n^{o(k)}$ algorithm

Is Clique a good source of hardness?

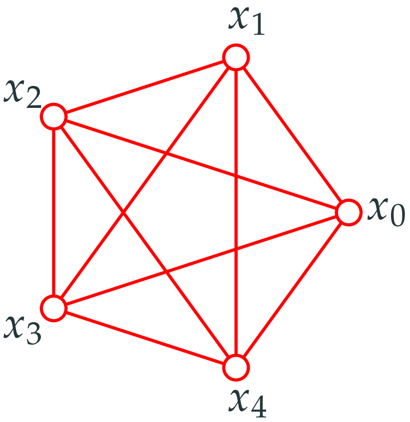


$x_i \in \Sigma_i = V(G)$

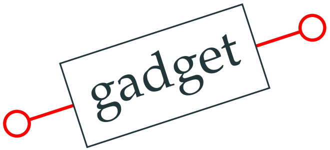


no $n^{o(k)}$ algorithm

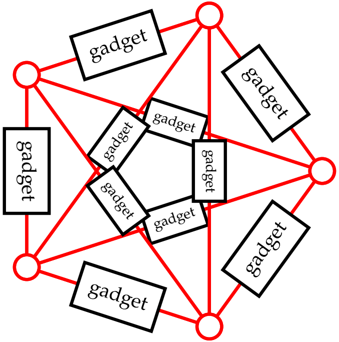
Is CLIQUE a good source of hardness?



no $n^{o(k)}$ algorithm

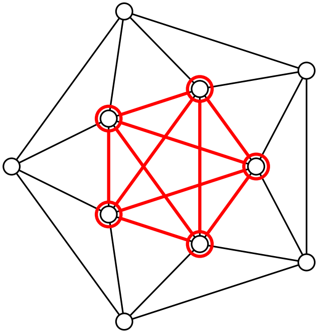


Is Clique a good source of hardness?

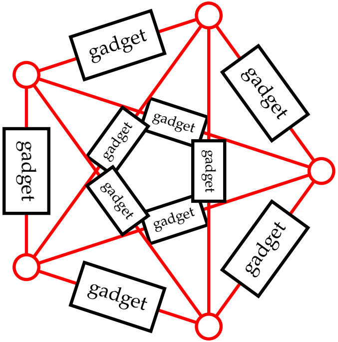


no $n^{o(k)}$ algorithm

Is Clique a good source of hardness?

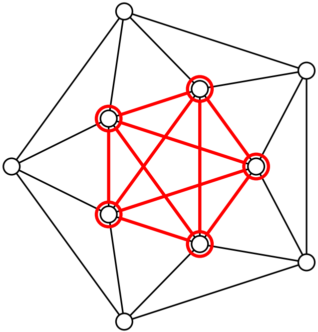


$t = \Theta(k^2)$
gadgets

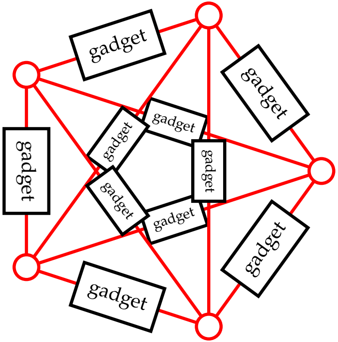


no $n^{o(k)}$ algorithm

Is Clique a good source of hardness?



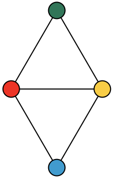
$t = \Theta(k^2)$
gadgets

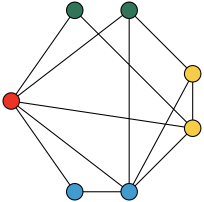


no $n^{o(k)}$ algorithm

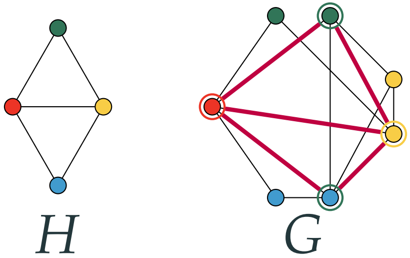
no $n^{o(\sqrt{t})}$ algorithm

colourful subgraph isomorphism ColSub(H)

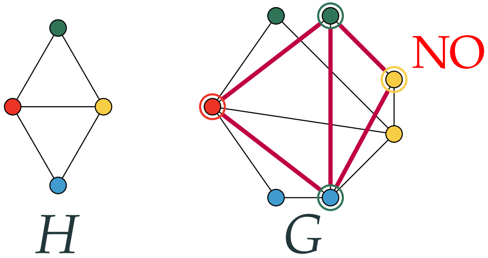




colourful subgraph isomorphism ColSub(H)



colourful subgraph isomorphism ColSub(H)



colourful subgraph isomorphism ColSub(H)

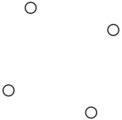
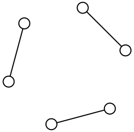
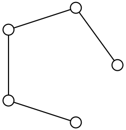
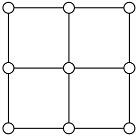
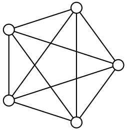
Do there exist **sparse** graphs H_ℓ of ℓ **edges** such that ColSub(H) cannot be solved in time $n^{o(\ell)}$?

colourful subgraph isomorphism ColSub(H)

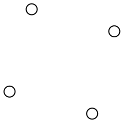
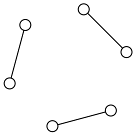
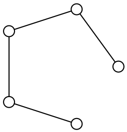
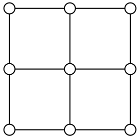
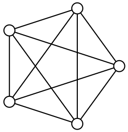
Do there exist **sparse** graphs H_ℓ of ℓ **edges** such that ColSub(H) cannot be solved in time $n^{o(\ell)}$?

If this is true, then we have **tight** lower bounds for:

colourful subgraph isomorphism ColSub(H)

indset	matching	path	grid	clique
				

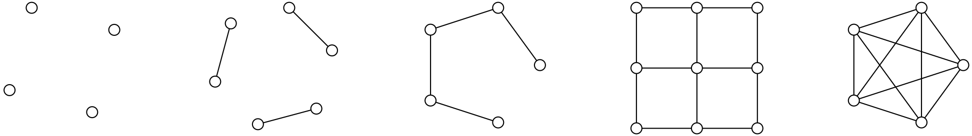
colourful subgraph isomorphism ColSub(H)

indset	matching	path	grid	clique
				

trivial

colourful subgraph isomorphism $\textsc{ColSub}(H)$

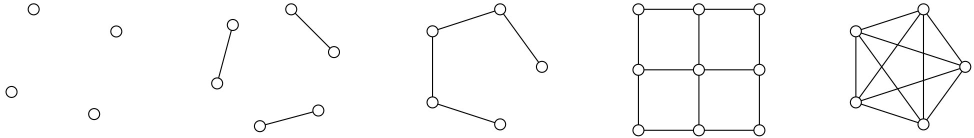
indset	matching	path	grid	clique



trivial	in **P**

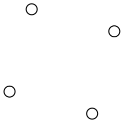
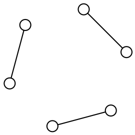
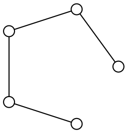
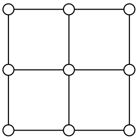
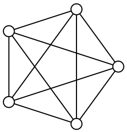
colourful subgraph isomorphism ColSub(H)

indset	matching	path	grid	clique

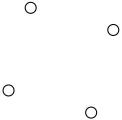
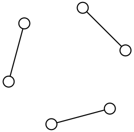
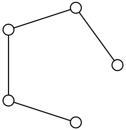
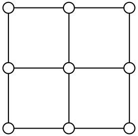
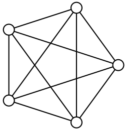


trivial	in **P**	**NP**-hard		

colourful subgraph isomorphism ColSub(H)

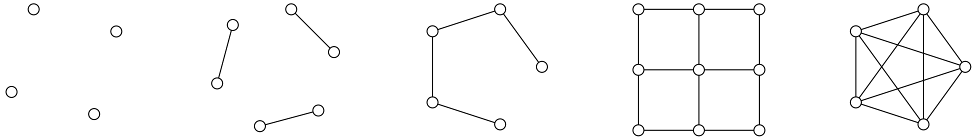
indset	matching	path	grid	clique
				
trivial	in **P**	**NP**-hard c^kpoly(n)		

colourful subgraph isomorphism ColSub(H)

indset	matching	path	grid	clique
				
trivial	in **P**	**NP**-hard $c^k \mathrm{poly}(n)$	**W[1]**-hard	

colourful subgraph isomorphism ColSub(H)

indset | matching | path | grid | clique



trivial

in **P**

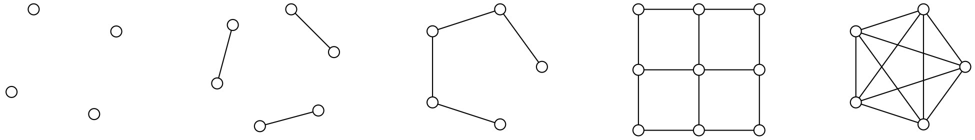
NP-hard

$c^k \text{poly}(n)$

W[1]-hard

$n^{\sqrt{k}}$

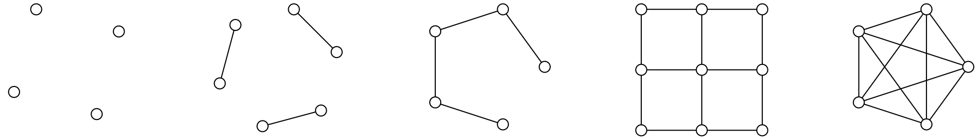
colourful subgraph isomorphism ColSub(H)



indset	matching	path	grid	clique
trivial	in **P**	**NP**-hard	**W[1]**-hard	**W[1]**-hard
		$c^k\mathrm{poly}(n)$	$n^{\sqrt{k}}$	

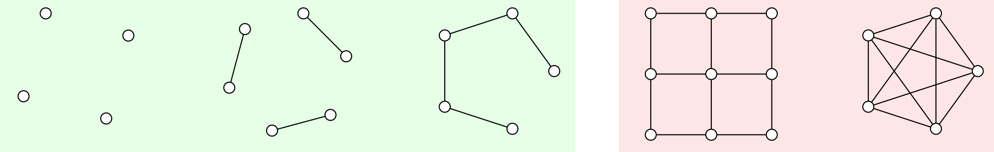
colourful subgraph isomorphism ColSub(H)

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'



colourful subgraph isomorphism ColSub(H)

indset	matching	path	grid	clique

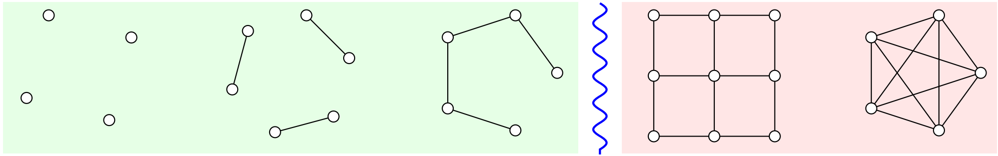


FPT **W[1]**-hard

trivial	in **P**	**NP**-hard	**W[1]**-hard	**W[1]**-hard
		$c^k \text{poly}(n)$	$n^{\sqrt{k}}$	'ETH-hard'

colourful subgraph isomorphism ColSub(H)

indset	matching	path	grid	clique

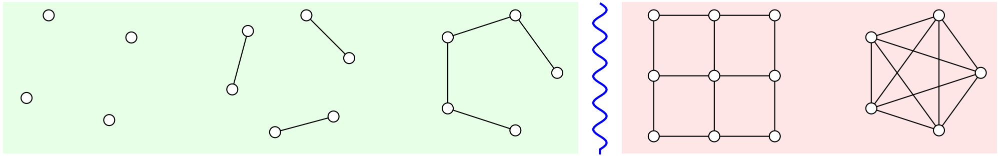


FPT ? **W[1]**-hard

indset	matching	path	grid	clique
trivial	in **P**	**NP**-hard	**W[1]**-hard	**W[1]**-hard
		$c^k \text{poly}(n)$	$n^{\sqrt{k}}$	‘ETH-hard’

large **treewidth** $\Longleftrightarrow$ **W[1]**-hardness

indset	matching	path	grid	clique

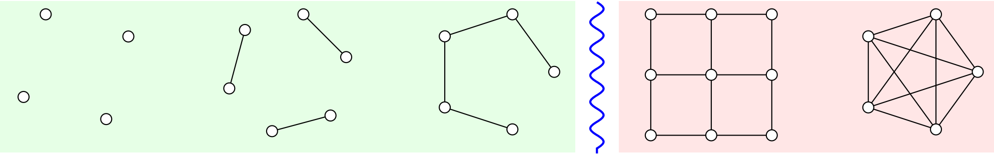


FPT ? **W[1]**-hard

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'

large **treewidth** $\iff$ **W[1]**-hardness

indset	matching	path	grid	clique

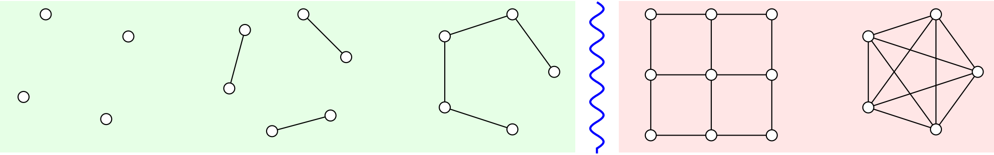


FPT ? **W[1]**-hard

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'

treewidth t implies $n^{\Omega(t/\log t)}$ lower bound

indset	matching	path	grid	clique



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

Theorem **[Marx'10]**

Unless ETH fails, ColSub(H) cannot be solved in time

$$f(H) \cdot n^{o\left(\frac{t}{\log t}\right)} \quad \text{where} \quad t = \mathrm{tw}(H).$$

treewidth t implies $n^{\Omega(t/\log t)}$ lower bound

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

treewidth t implies $n^{\Omega(t/\log t)}$ lower bound

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

Explicit construction of sparse expanders

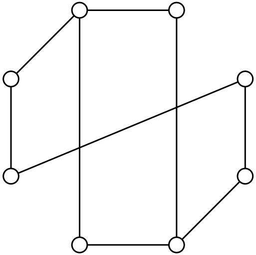
Expanders have linear treewidth

Theorem

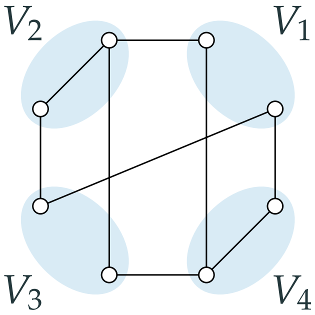
[Marx'10]

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.

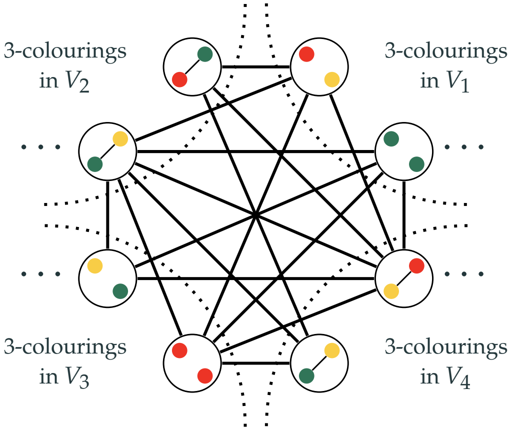
3-Colouring instance



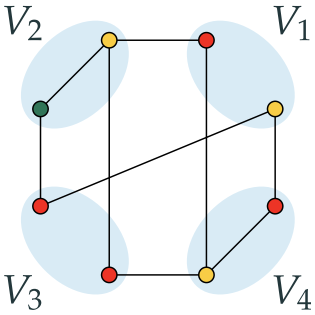
3-Colouring instance



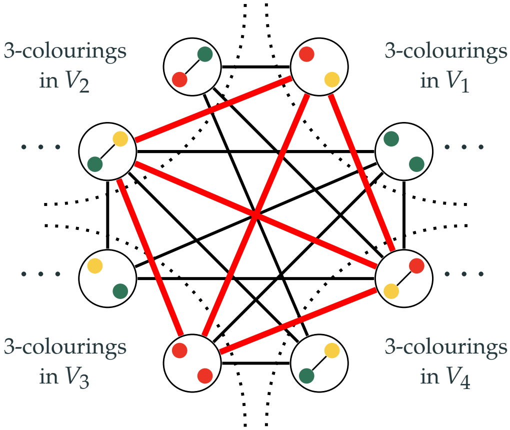
k-Clique instance



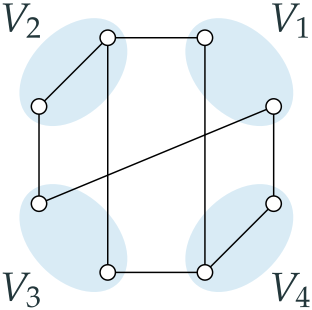
3-Colouring instance



k-Clique instance

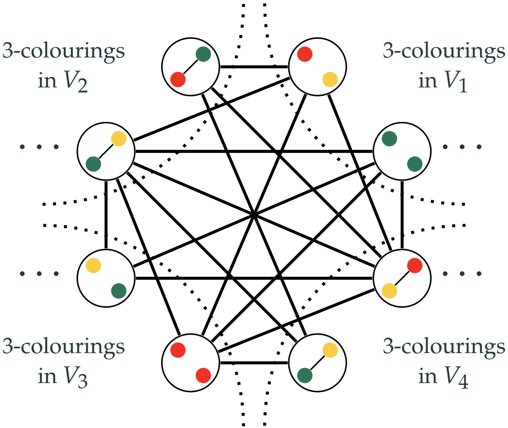


3-Colouring instance



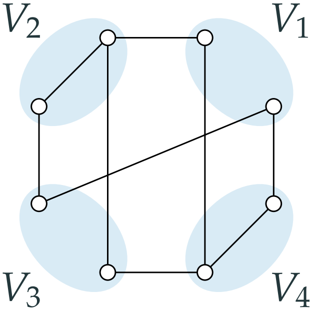
n vertices
k parts

k-Clique instance



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

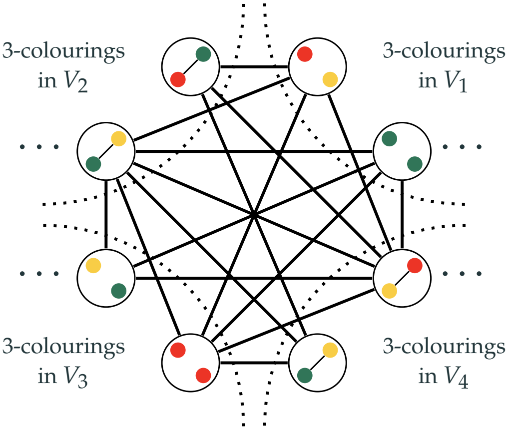
3-Colouring instance



n vertices
k parts

$N^{o(k)}$

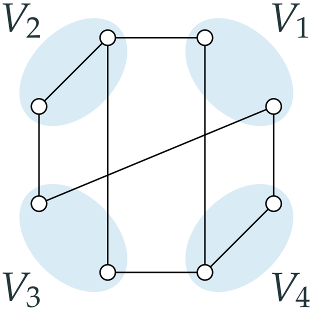
k-Clique instance



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

$2^{o(n)}$

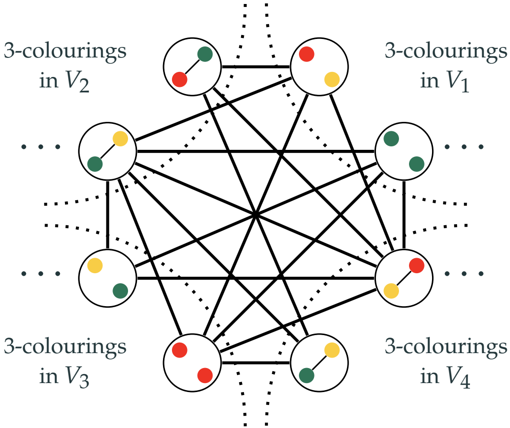
3-Colouring instance



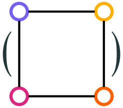
n vertices
k parts

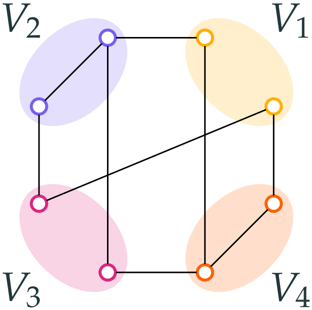
$N^{o(k)}$

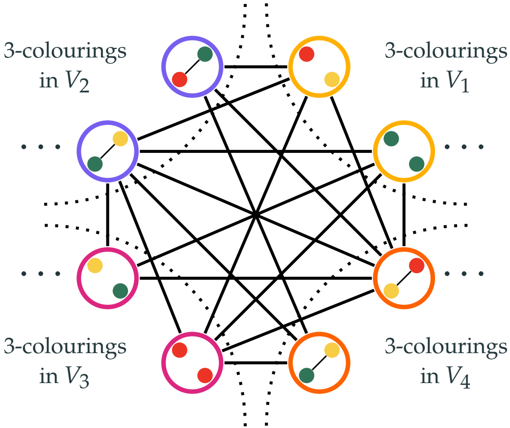
k-Clique instance

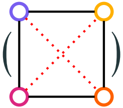


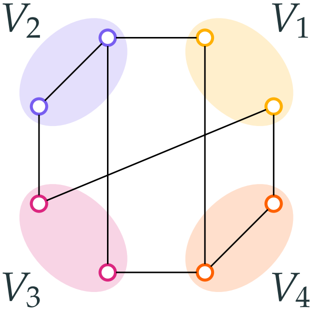
$N = k \cdot 3^{n/k}$ vertices

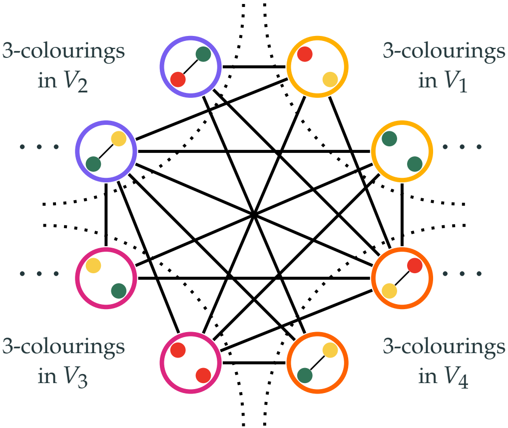
3-Colouring $\preceq$ ColSub ()

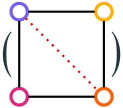


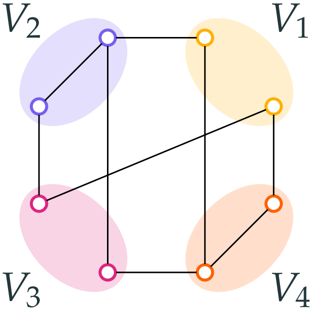


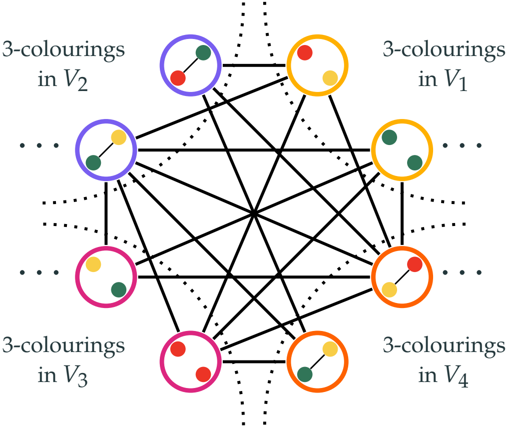
3-Colouring $\preceq$ ColSub ()

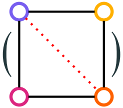


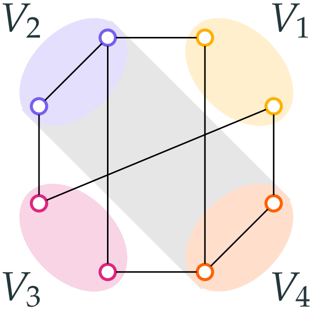


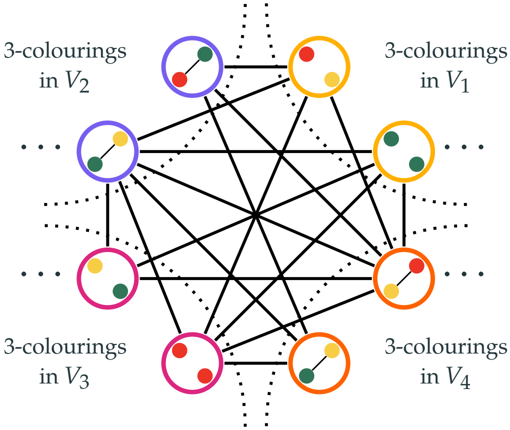
3-Colouring $\preceq$ ColSub ()

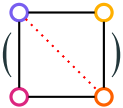


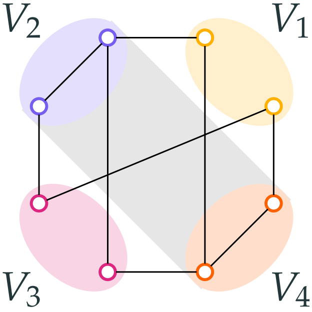


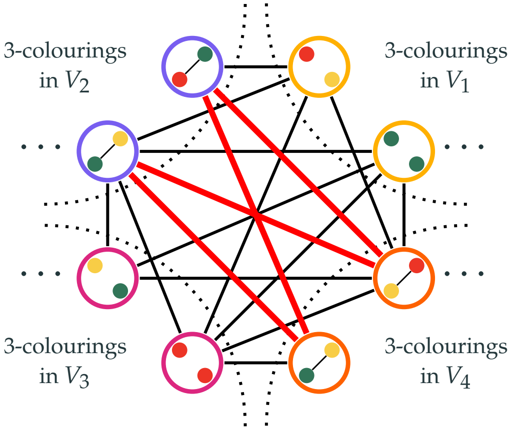
3-COLOURING $\preceq$ COLSUB ()

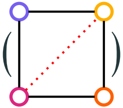


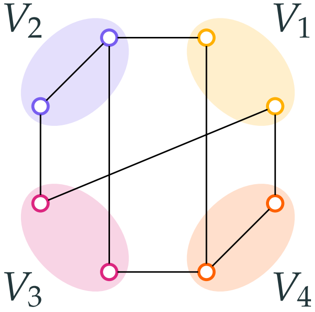


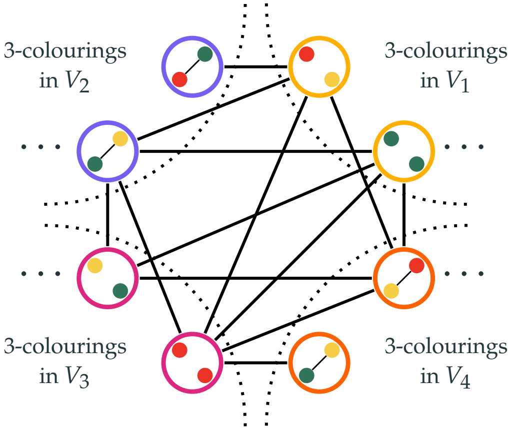
3-Colouring $\preceq$ ColSub ()

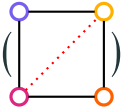


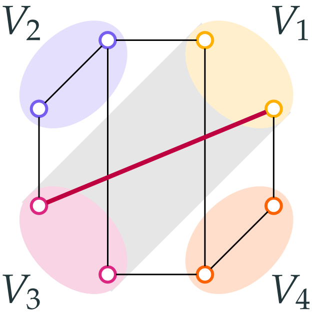


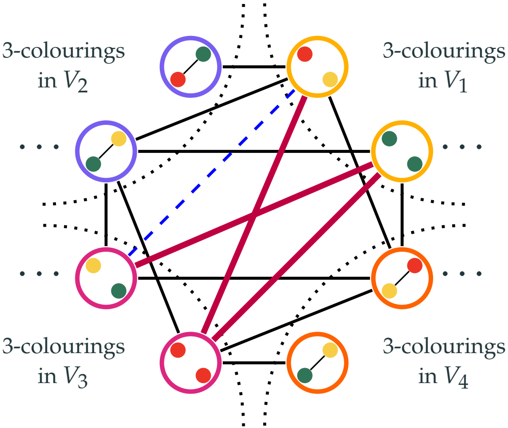
3-Colouring $\preceq$ ColSub 

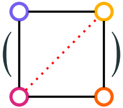


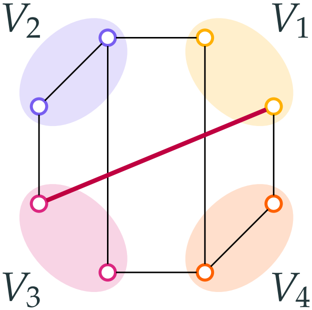


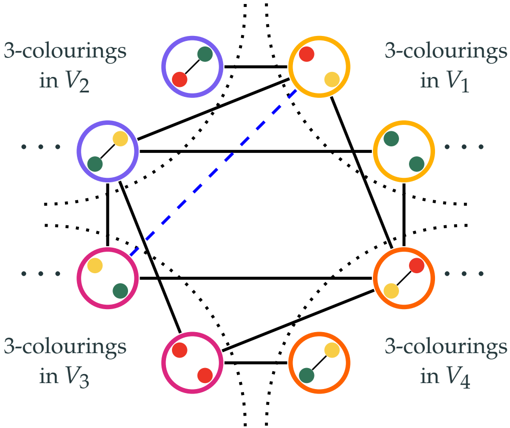
3-COLOURING $\preceq$ COLSUB ()

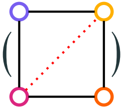


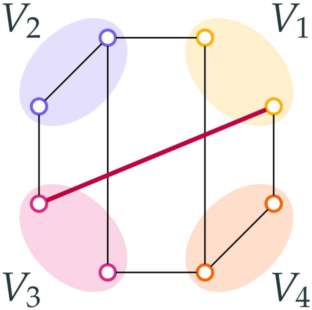


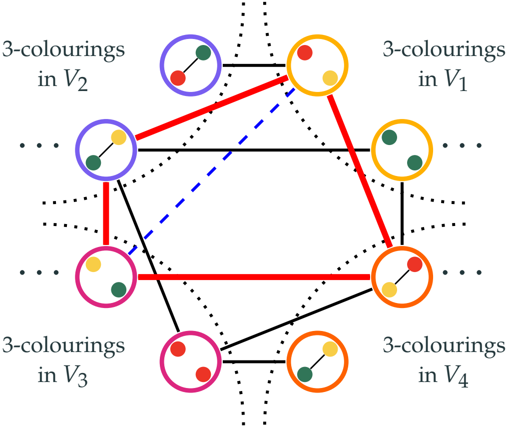
3-Colouring $\preceq$ ColSub ()

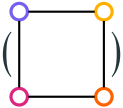


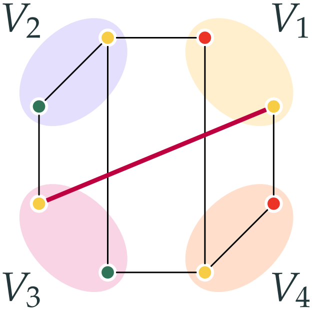


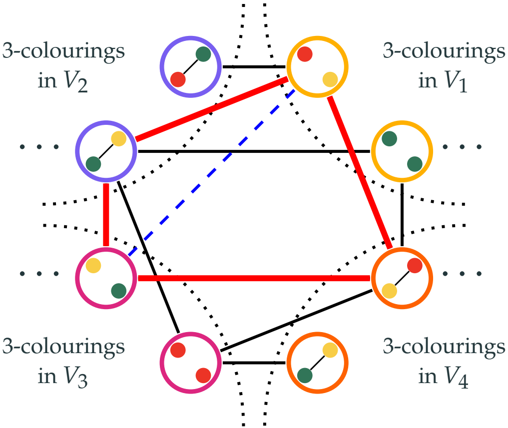
3-COLOURING $\preceq$ COLSUB ()

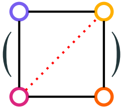


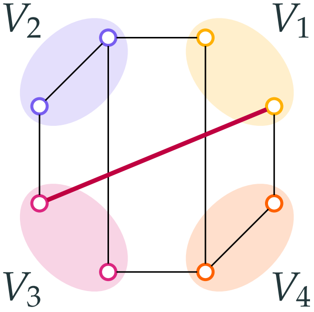


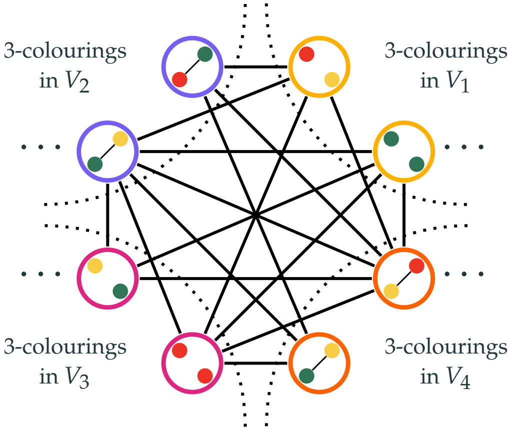
3-Colouring $\preceq$ ColSub ()

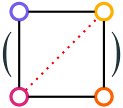


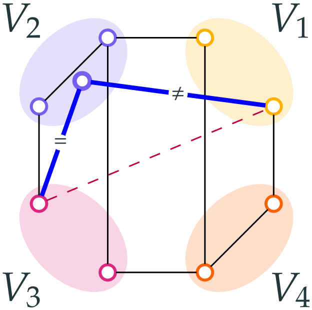


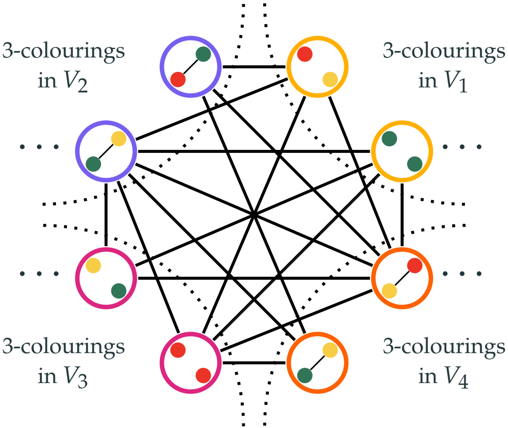
3-Colouring $\preceq$ ColSub ()

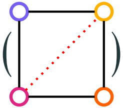


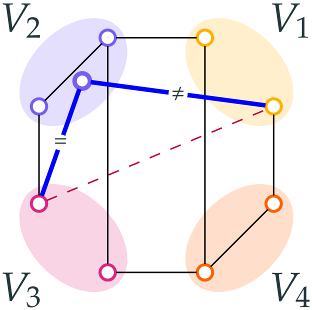


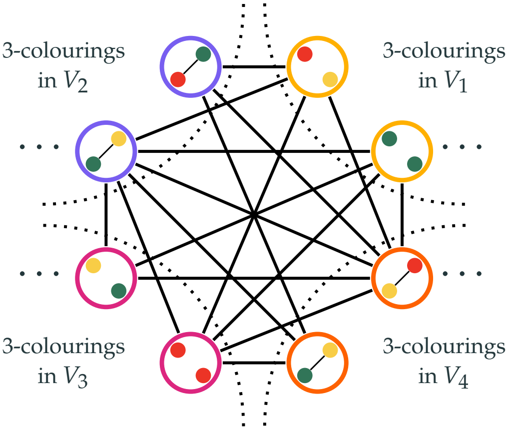
3-Colouring $\preceq$ ColSub ()



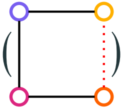


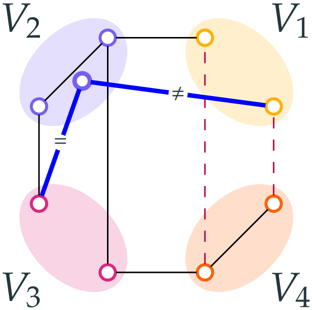
3-COLOURING $\preceq$ COLSUB ()

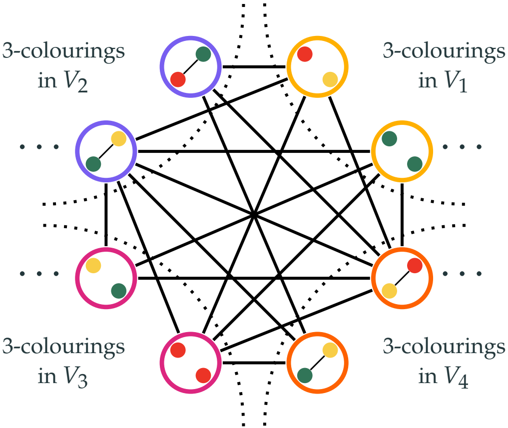




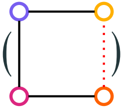
But this costs us something...

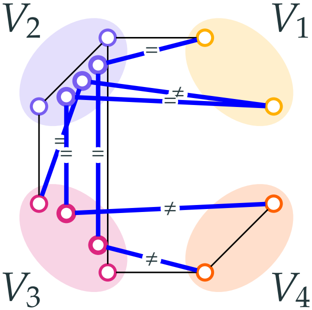
3-Colouring $\preceq$ ColSub 

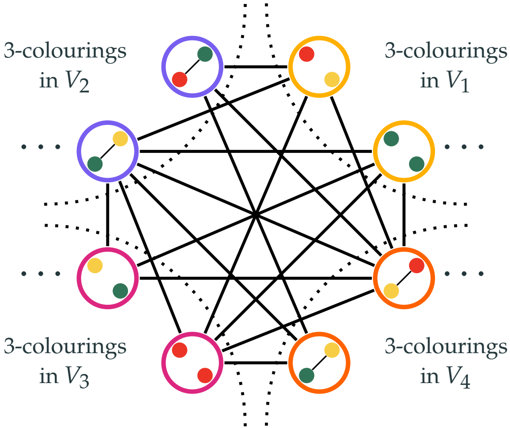




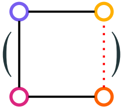
But this costs us something...

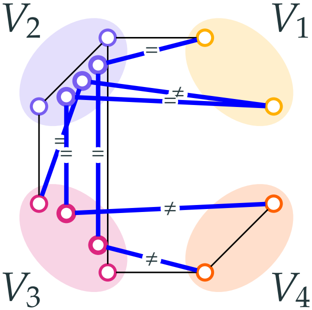
3-Colouring $\preceq$ ColSub ()

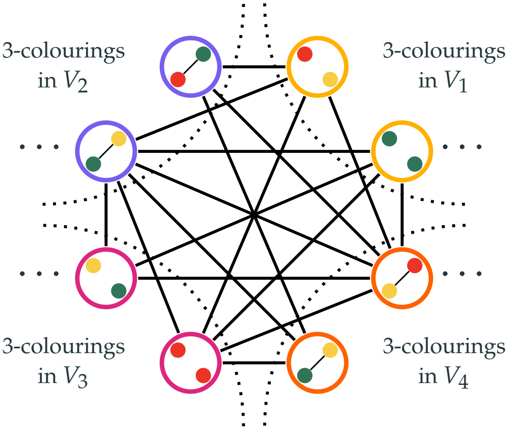




But this costs us something...

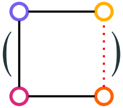
3-Colouring ≤ ColSub ()

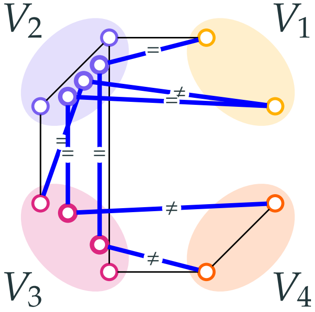


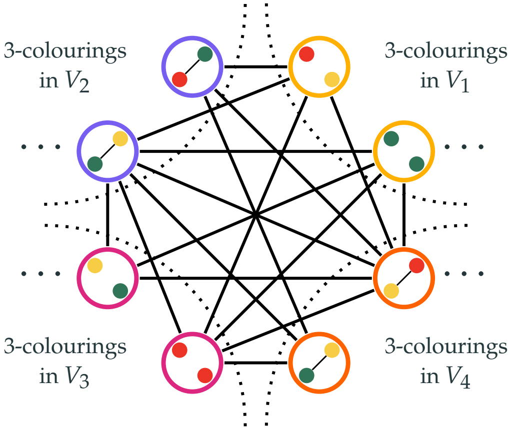


But this costs us something... Too many new vertices in V_2!

#vertices in V_i
$\gg n/k$
#configs = N
$\gg 3^{n/k}$

3-Colouring $\preceq$ ColSub ()

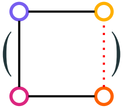


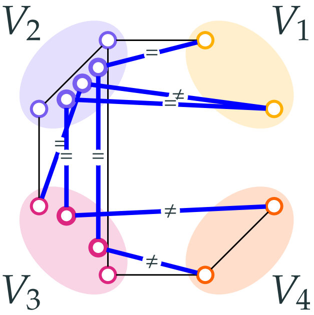


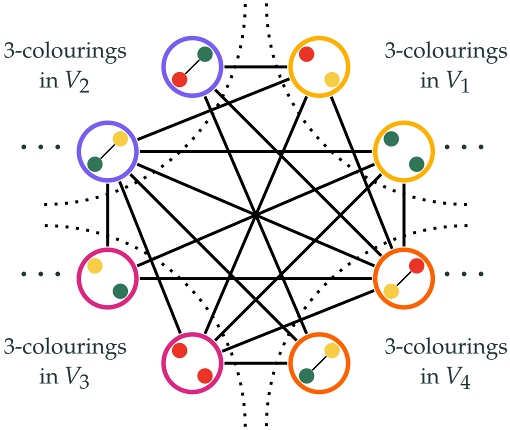
But this costs us something... Too many new vertices in V_2!

#vertices in V_i $\gg n/k$

#configs = N $\gg 3^{n/k}$

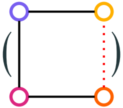
3-Colouring $\preceq$ ColSub ()

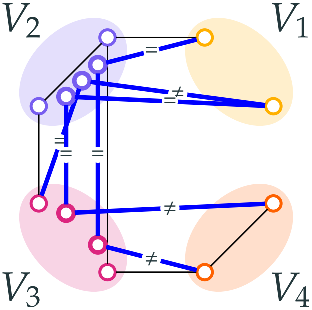


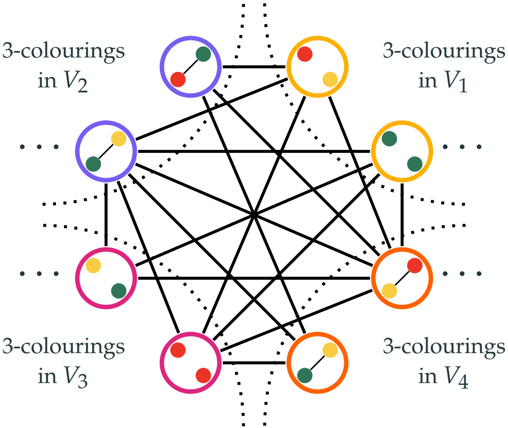


Routing in paths are highly **congested**!

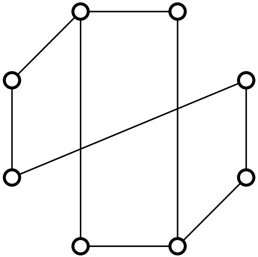
#vertices in V_i
$\gg n/k$
#configs $= N$
$\gg 3^{n/k}$

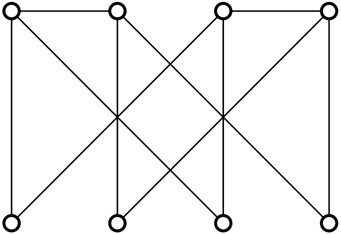
3-Colouring $\preceq$ ColSub()



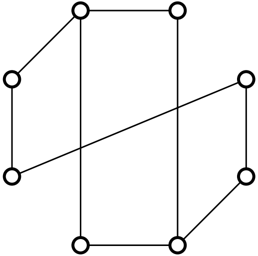


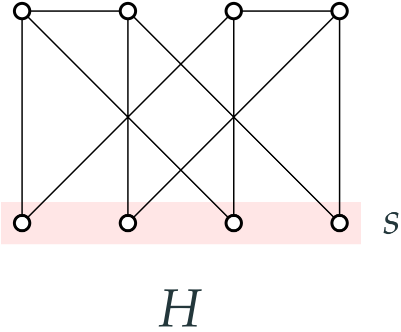
Routing in paths are highly **congested**! Indeed, ColSub(path) is FPT.





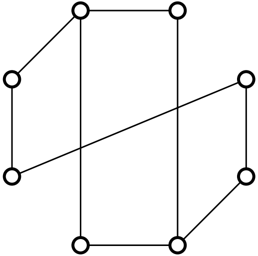
H

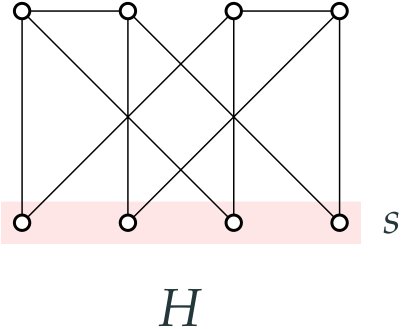




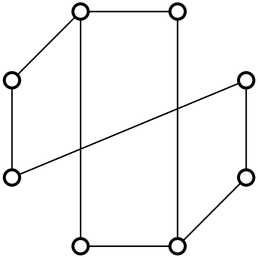
s
H

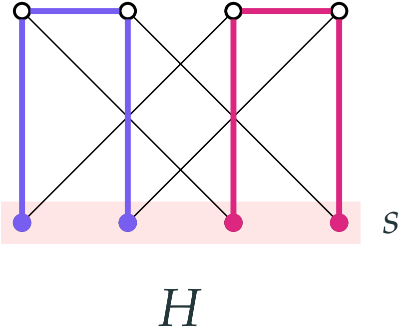
matching-linked set



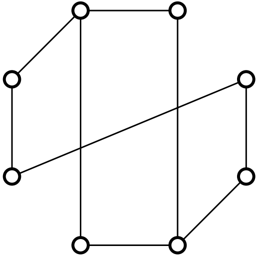


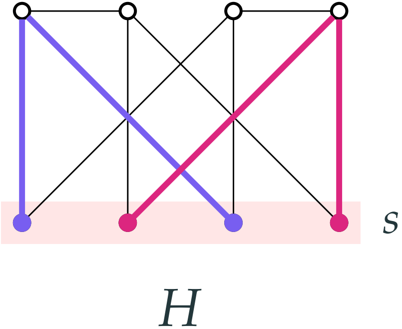
matching-linked set



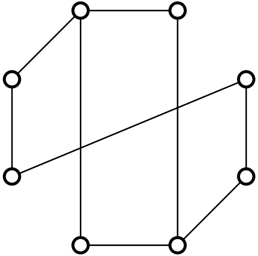


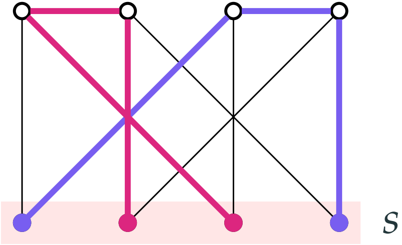
matching-linked set





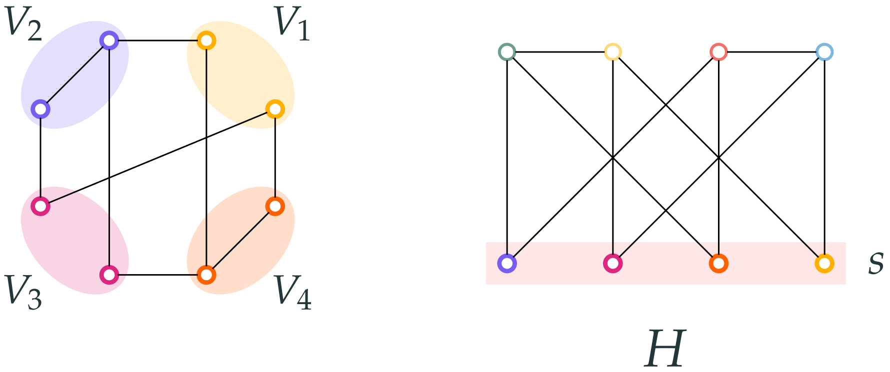
matching-linked set



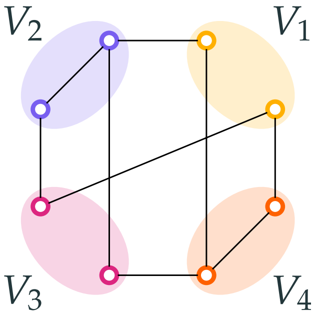


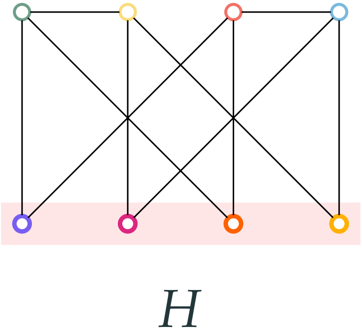
H

matching-linked set

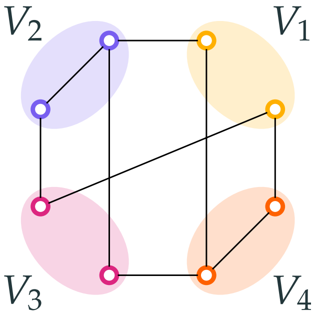


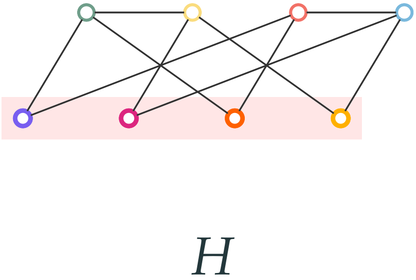
matching-linked set



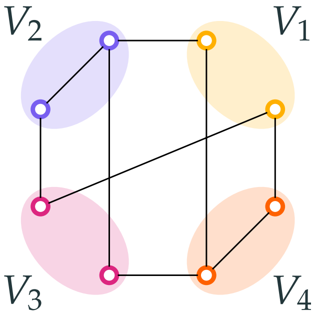


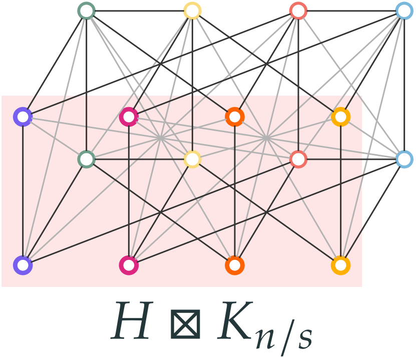
matching-linked set



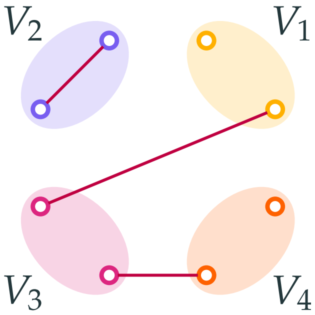


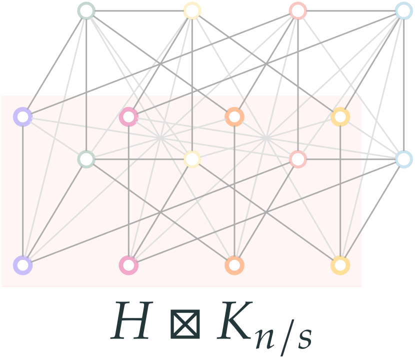
matching-linked set in the **blow-up** graph



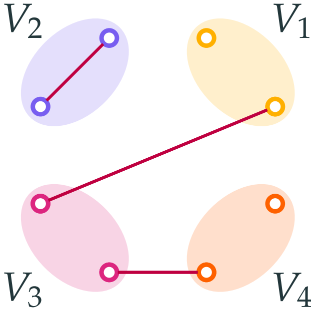


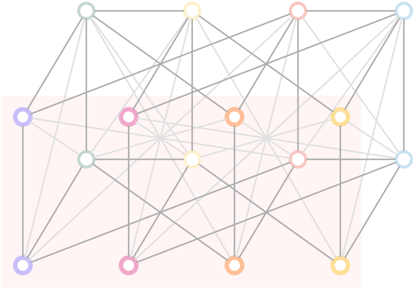
matching-linked set in the **blow-up** graph





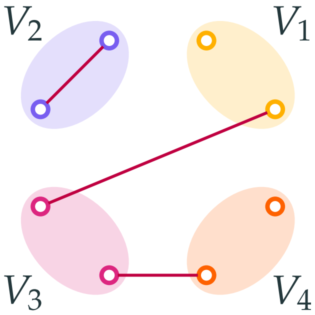
matching-linked set in the **blow-up** graph

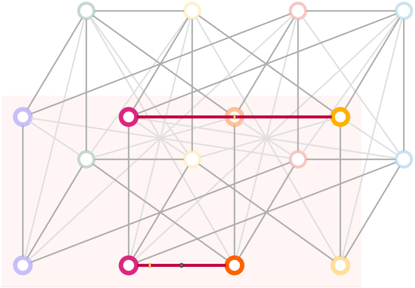




$H \boxtimes K_{n/s}$

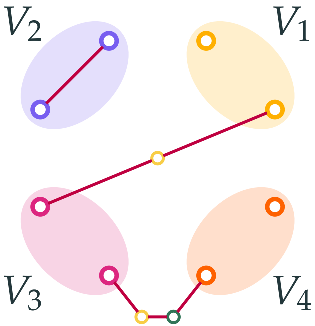
matching-linked set in the **blow-up** graph

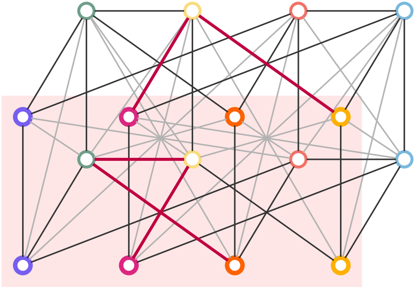




$H \boxtimes K_{n/s}$

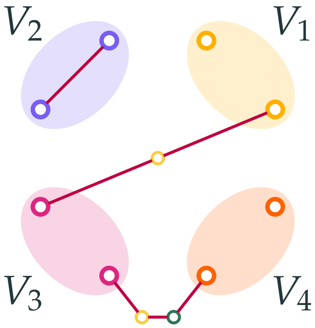
matching-linked set in the **blow-up** graph

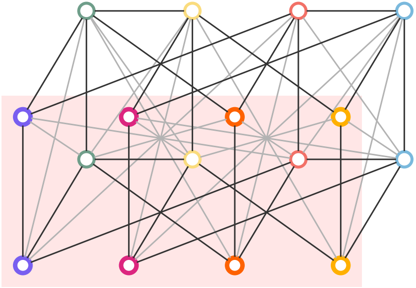




$H \boxtimes K_{n/s}$

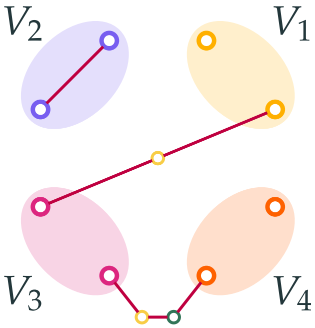
matching-linked set in the **blow-up** graph



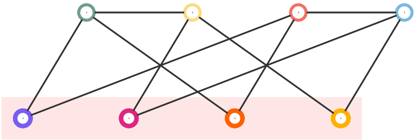


$H \boxtimes K_{n/s}$

matching-linked set

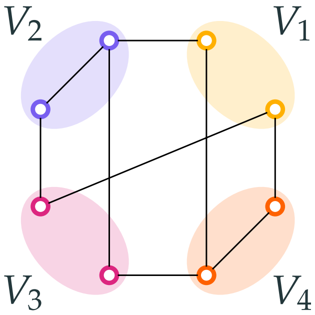


#vertices in each colour $\leq n/s$

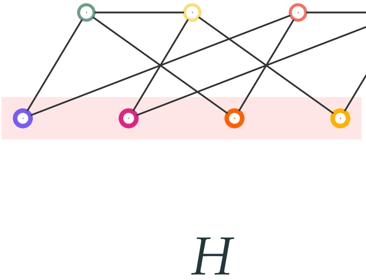


H

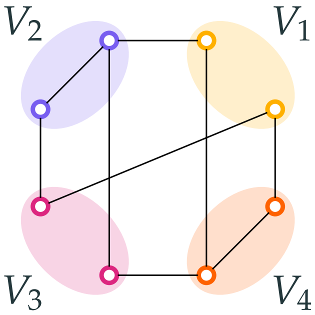
matching-linked set



#vertices in each colour $\leq 5n/s$

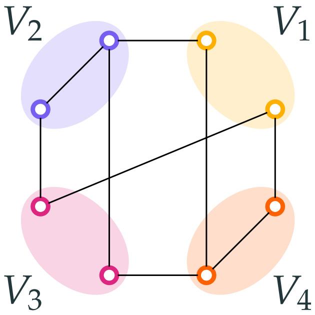


matching-linked set



#config vertices $N \leq k \cdot 3^{5n/s}$

matching-linked set



#config vertices $N \leq k \cdot 3^{5n/s}$

$s = k/g(k)$ gives $N^{k/g(k)}$

lower bound

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 ColSub(H_ℓ) cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.

It suffices to find a graph H that

(1) has $k = O(s \log s)$ 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 $k = O(s \log s)$ 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 Labs 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 $k = O(s \log s)$ 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 Labs in 1964

Fun fact: it is NOT an expander.

(**[Marx'10]** and its subsequential simplification **[C.S.-Marx-Pilipczuk-Souza'24]** essentially require expanders)

Theorem

[Marx'10]

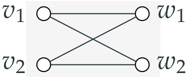
There is a sequence of **degree-4** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and $\textsc{ColSub}(H_\ell)$ cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.

Theorem

[Marx'10]

There is a sequence of **degree-4** 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.

$B_1 =$

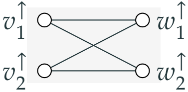


Theorem

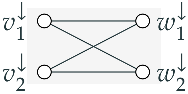
[Marx'10]

There is a sequence of **degree-4** 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.

$B_1^{\uparrow} =$



$B_1^{\downarrow} =$



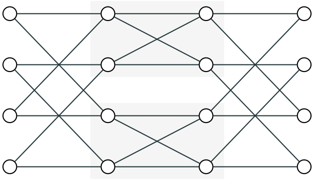
Theorem

[Marx'10]

There is a sequence of **degree-4** 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.

$B_1^{\uparrow} =$

$B_1^{\downarrow} =$

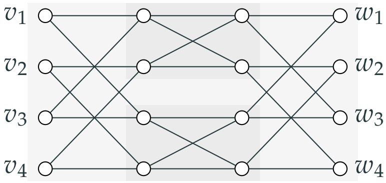


Theorem

[Marx'10]

There is a sequence of **degree-4** 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.

$B_2 =$

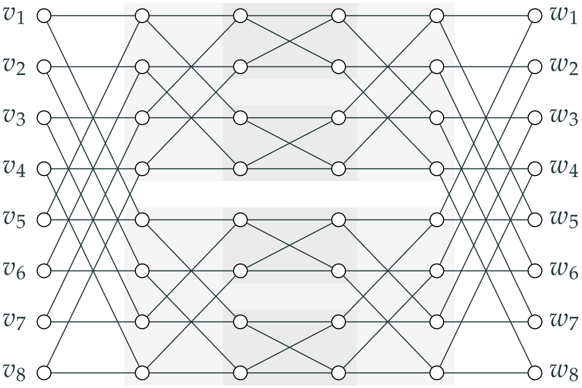


Theorem

[Marx'10]

There is a sequence of **degree-4** 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.

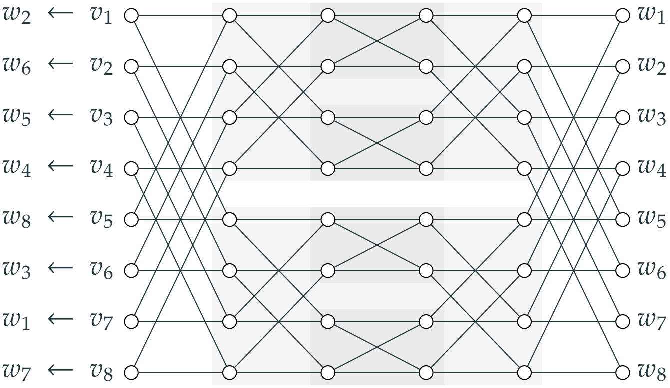
$B_3 =$



Theorem

[Marx'10]

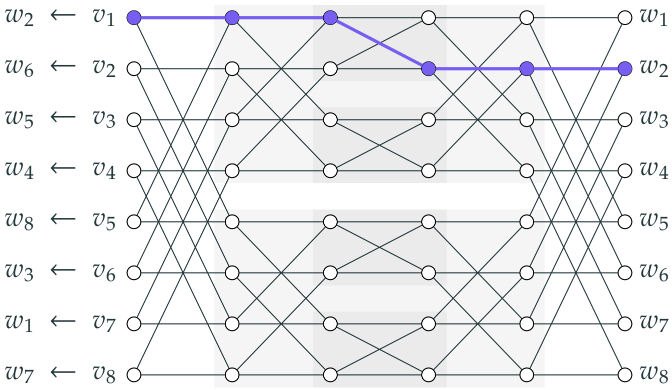
There is a sequence of **degree-4** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and ColSub(H_ℓ) cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.



Theorem

[Marx'10]

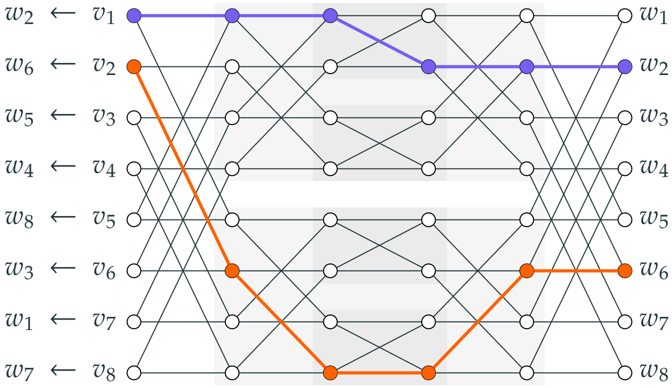
There is a sequence of **degree-4** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and ColSub(H_ℓ) cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.



Theorem

[Marx'10]

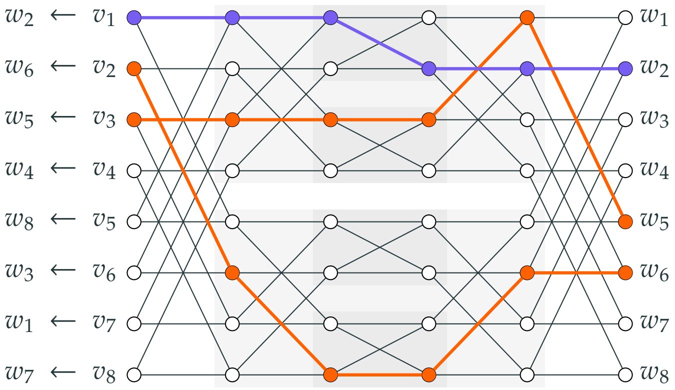
There is a sequence of **degree-4** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and ColSub(H_ℓ) cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.



Theorem

[Marx'10]

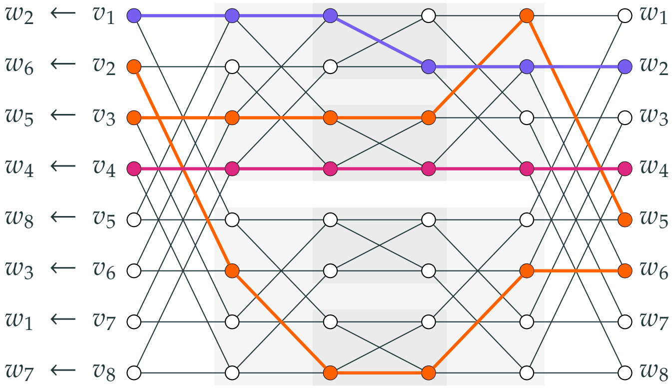
There is a sequence of **degree-4** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and $\textsc{ColSub}(H_\ell)$ cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.



Theorem

[Marx'10]

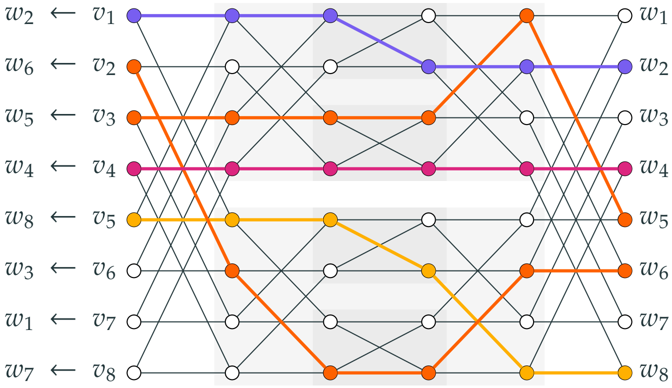
There is a sequence of **degree-4** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and ColSub(H_ℓ) cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.



Theorem

[Marx'10]

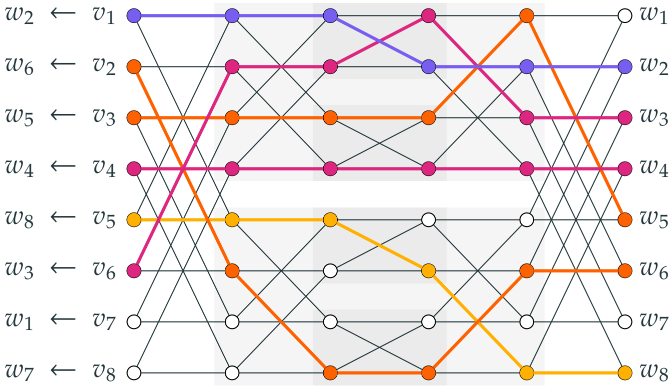
There is a sequence of **degree-4** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and ColSub(H_ℓ) cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.



Theorem

[Marx'10]

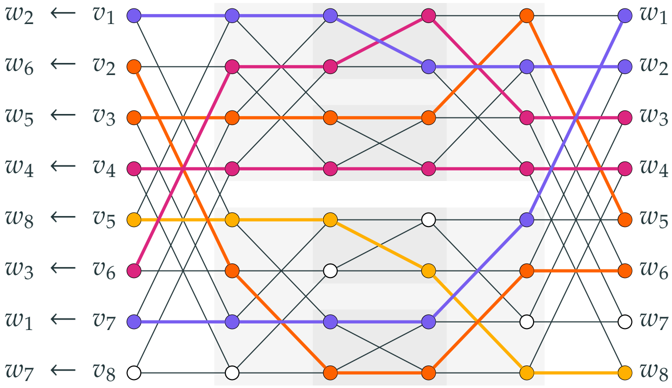
There is a sequence of **degree-4** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and ColSub(H_ℓ) cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.



Theorem

[Marx'10]

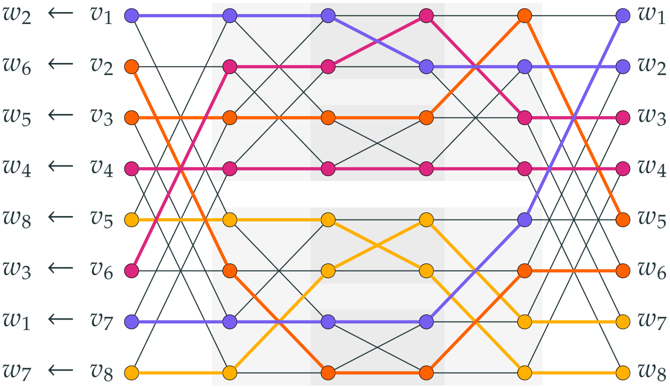
There is a sequence of **degree-4** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and ColSub(H_ℓ) cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.



Theorem

[Marx'10]

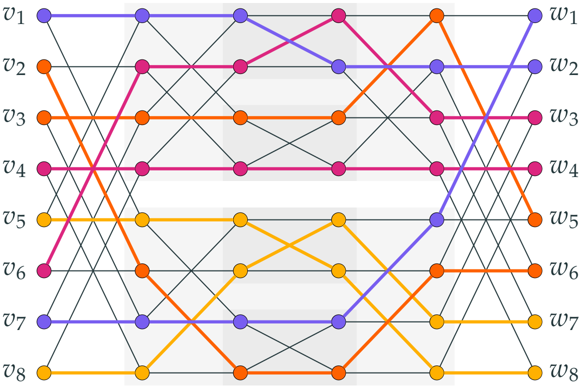
There is a sequence of **degree-4** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and ColSub(H_ℓ) cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.



Theorem

[Marx'10]

There is a sequence of **degree-4** 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.

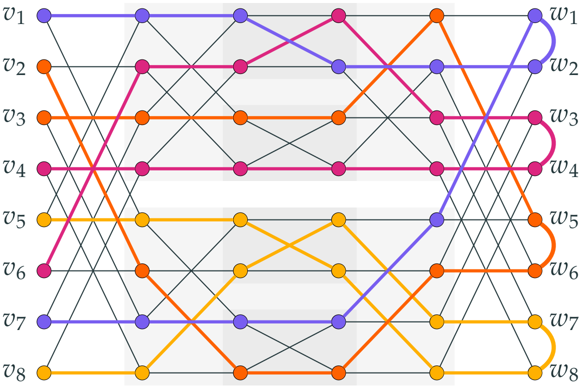


Link up $M = \{v_1v_7, v_2v_3, v_4v_6, v_5v_8\}$?

Theorem

[Marx'10]

There is a sequence of **degree-4** 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.

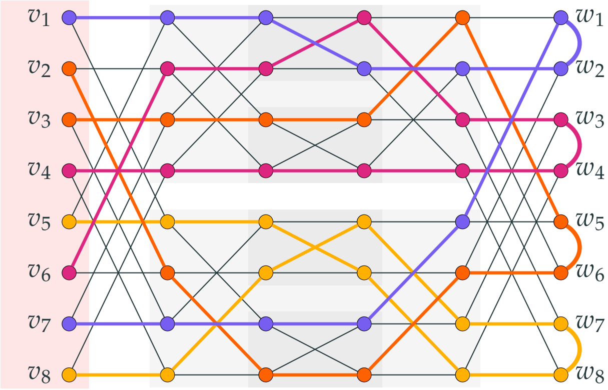


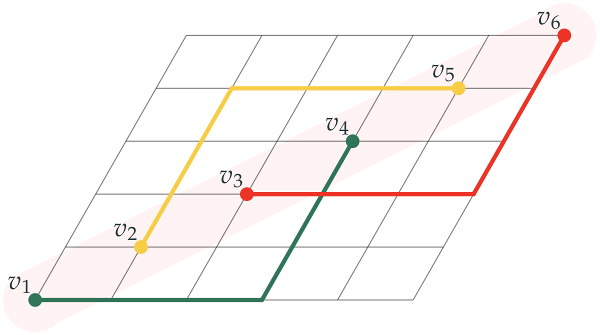
Link up $M = \{v_1v_7, v_2v_3, v_4v_6, v_5v_8\}$?

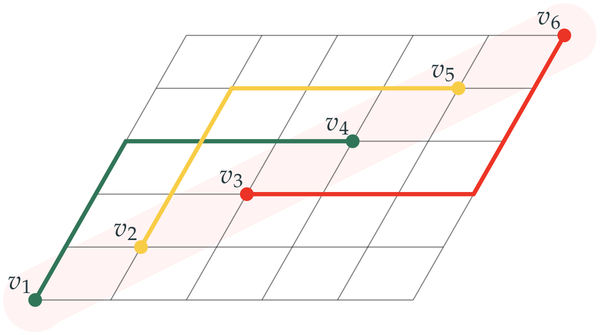
Theorem

[Marx'10]

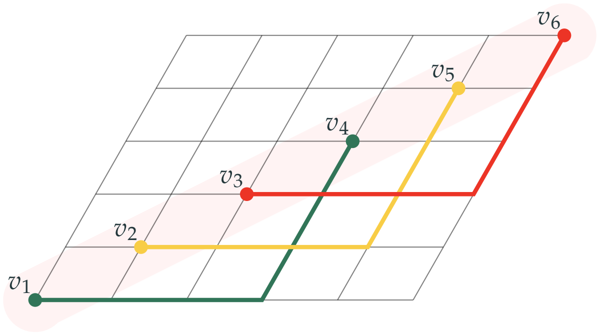
There is a sequence of **degree-4** graphs $H_1, H_2, \cdots$ s.t. H_ℓ has ℓ edges and ColSub(H_ℓ) cannot be solved in time $n^{o(\ell/\log \ell)}$ unless ETH fails.



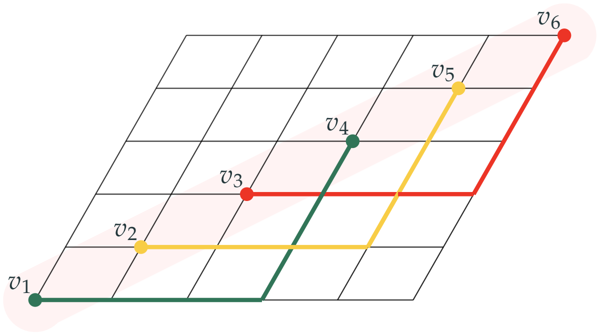

v_1
v_2
v_3
v_4
v_5
v_6



v_6
v_5
v_4
v_3
v_2
v_1

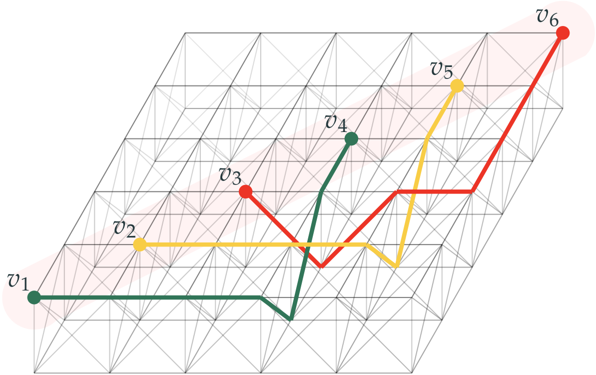


v_1
v_2
v_3
v_4
v_5
v_6

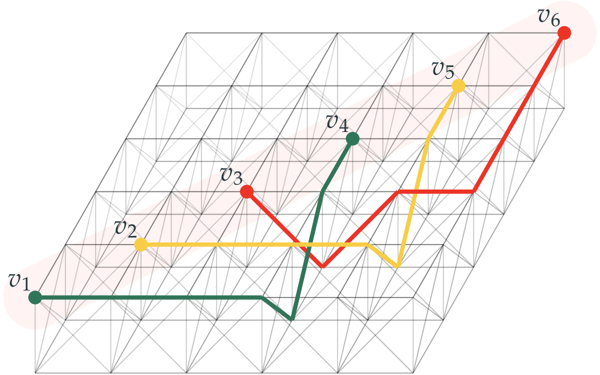


v_1
v_2
v_3
v_4
v_5
v_6

$\gamma(H)$: **linkage capacity** $\approx \dfrac{\max \big|\text{matching-linked set}\big| \text{ w. congestion}}{\text{congestion}}$



$\gamma(H)$: **linkage capacity** $\approx \dfrac{\max \left|\text{matching-linked set}\right| \text{ w. congestion}}{\text{congestion}}$



For **any** graph H, no $n^{o(\gamma(H))}$ algorithm for ColSub(H) unless ETH fails.

$$\gamma(H)\text{: }\textbf{linkage capacity} \approx \frac{\max \left|\text{matching-linked set}\right| \text{ w. congestion}}{\text{congestion}}$$

Unless ETH fails, ColSub(H) cannot be solved in time

$$\gamma(H)\text{: } \textbf{linkage capacity} \approx \frac{\max \left|\text{matching-linked set}\right| \text{ w. congestion}}{\text{congestion}}$$

Unless ETH fails, ColSub(H) cannot be solved in time

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

$\gamma(H)$: **linkage capacity** $\approx \dfrac{\max \left|\text{matching-linked set}\right| \text{ w. congestion}}{\text{congestion}}$

Unless ETH fails, ColSub(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.

$\gamma(H)$: **linkage capacity** $\approx \dfrac{\max \big|\text{matching-linked set}\big| \text{ w. congestion}}{\text{congestion}}$

Unless ETH fails, ColSub(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.
- $n^{o(t/\log t)}$, for **any** graph with **treewidth** $t = \mathrm{tw}(H)$.
  - New proof to Marx's "Can you beat treewidth?" theorem.

$$\gamma(H)\text{: } \textbf{linkage capacity} \approx \frac{\max \left|\text{matching-linked set}\right| \text{ w. congestion}}{\text{congestion}}$$

Unless ETH fails, $\textsc{ColSub}(H)$ 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.
- $n^{o(t/\log t)}$, for **any** graph with **treewidth** $t = \mathrm{tw}(H)$.
  - New proof to Marx's "Can you beat treewidth?" theorem.

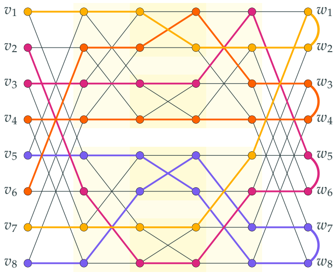
Implications to *induced subgraph counting*.

[Roth–Schmitt–Wellnitz'20, Döring–Marx–Wellnitz'24,25, Curticapean–Neuen'25]

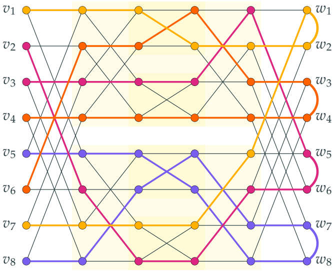
Summary

Summary

Hardness of subgraph counting via **linkage**.



Summary



Hardness of subgraph counting via **linkage**.

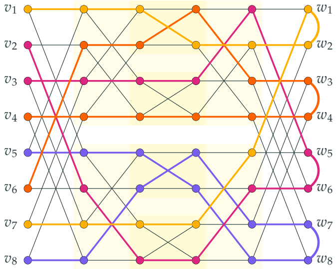
Beneš network for $n^{\Omega(k/\log k)}$ lower bound.

Summary

Hardness of subgraph counting via **linkage**.

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

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



Open problems

Open problems

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

Open problems

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?)

Open problems

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

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

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

Open problems

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?)

Novel usage of communication networks in complexity theory?

- extension complexity **[Göös–Jain–Watson'18]**
- PCP **[Bafna–Minzer–Vyas–Yun'25]**.

Open problems

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?)

Novel usage of communication networks in complexity theory?

- extension complexity **[Göös–Jain–Watson'18]**
- PCP **[Bafna–Minzer–Vyas–Yun'25]**.

New proofs of $(t/\log t)$-like lower bounds in other settings?

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

Open problems

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?)

Thank you!

Novel usage of communication networks in complexity theory?

- extension complexity **[Göös–Jain–Watson'18]**
- PCP **[Bafna–Minzer–Vyas–Yun'25]**.

New proofs of $(t/\log t)$-like lower bounds in other settings?

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

Bonus slides

formal proof of Beneš network property

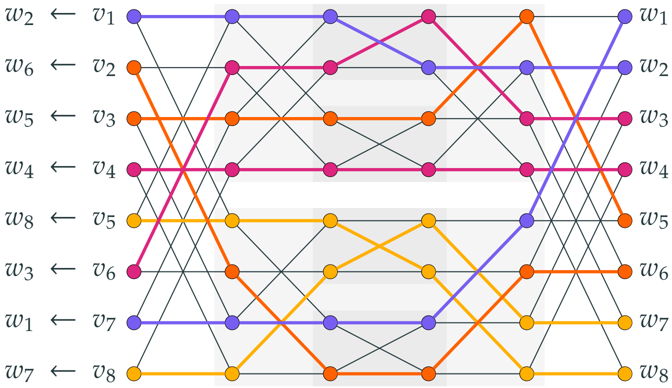
`https://tinyurl.com/benesnet`

thank Marcelo Mutzbauer for the amazing **Interactive Proof**

Theorem

[Beneš'1964]

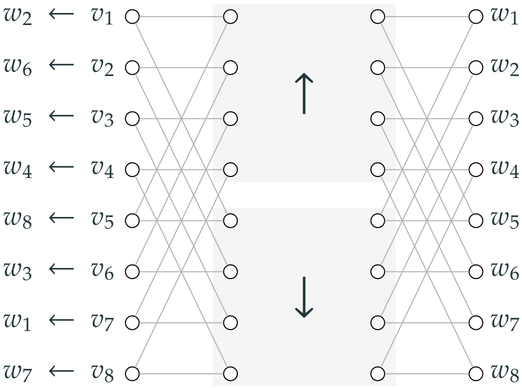
For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.



Theorem

[Beneš'1964]

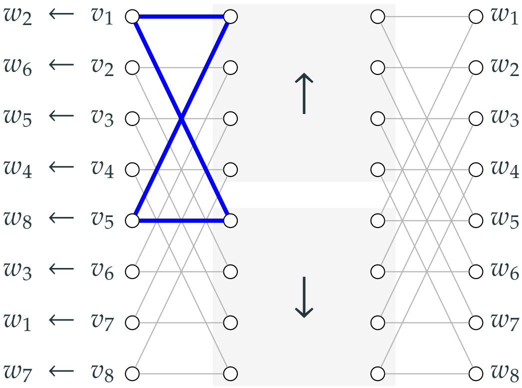
For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.



Theorem

[Beneš'1964]

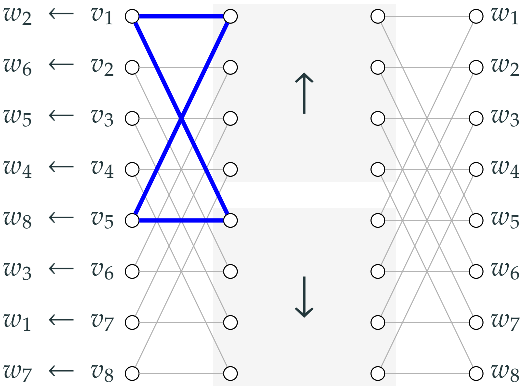
For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.

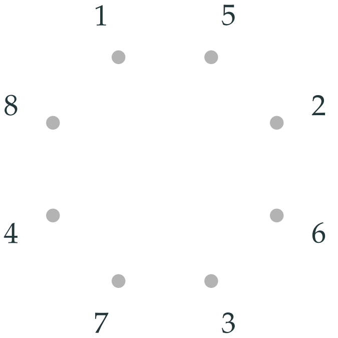


Theorem

[Beneš'1964]

For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.

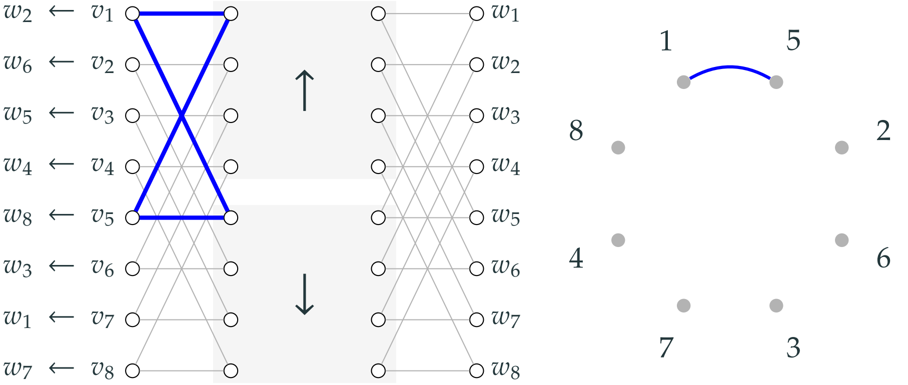




Theorem

[Beneš'1964]

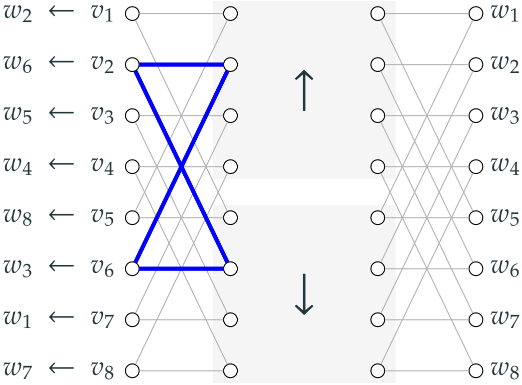
For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.

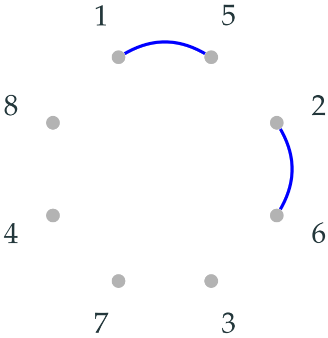


Theorem

[Beneš'1964]

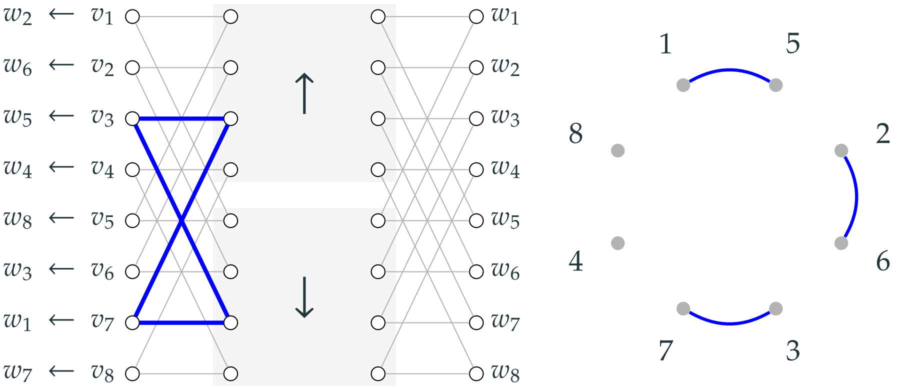
For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.





Theorem [Beneš'1964]

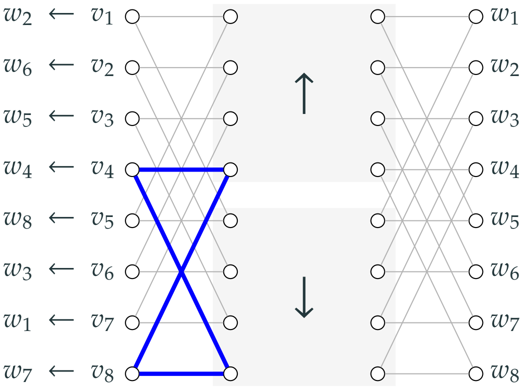
For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.

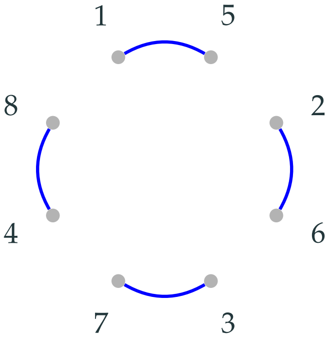


Theorem

[Beneš'1964]

For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.

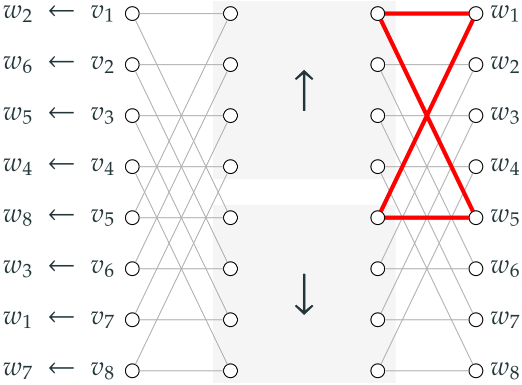


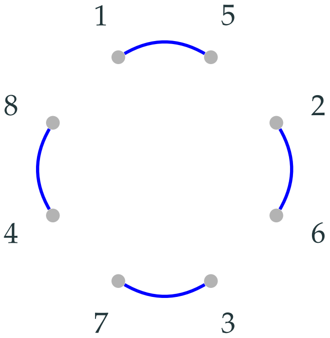


Theorem

[Beneš'1964]

For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.

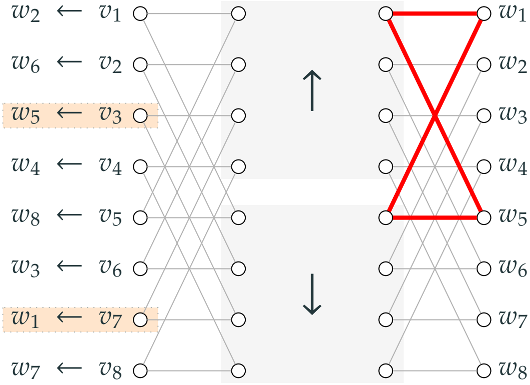


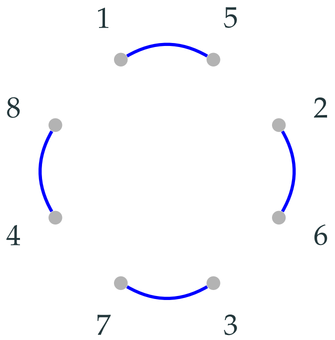


Theorem

[Beneš'1964]

For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.

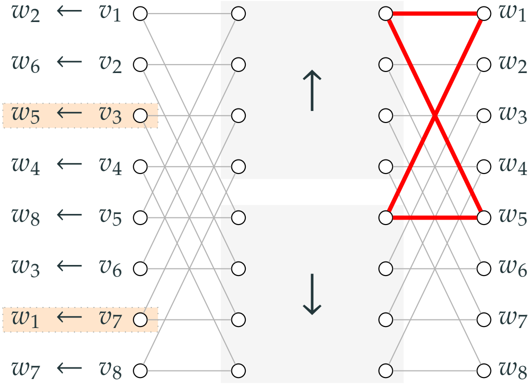


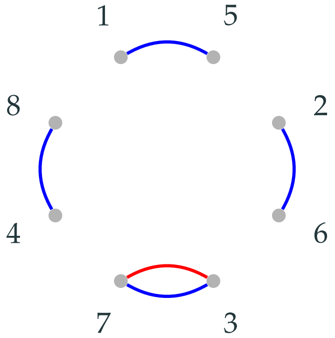


Theorem

[Beneš'1964]

For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.

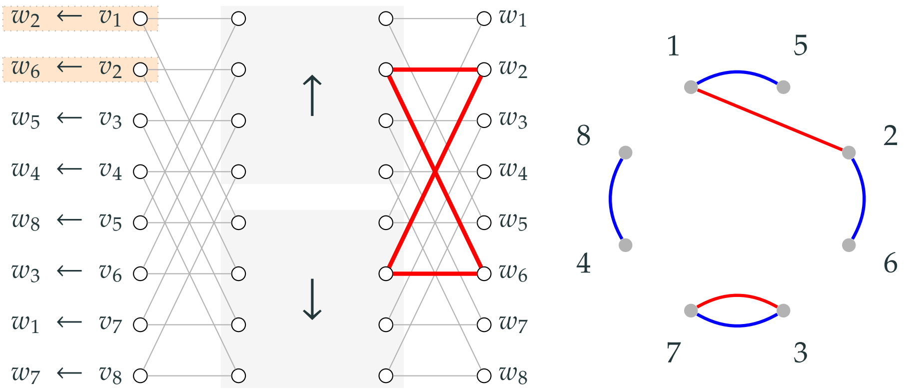




Theorem

[Beneš'1964]

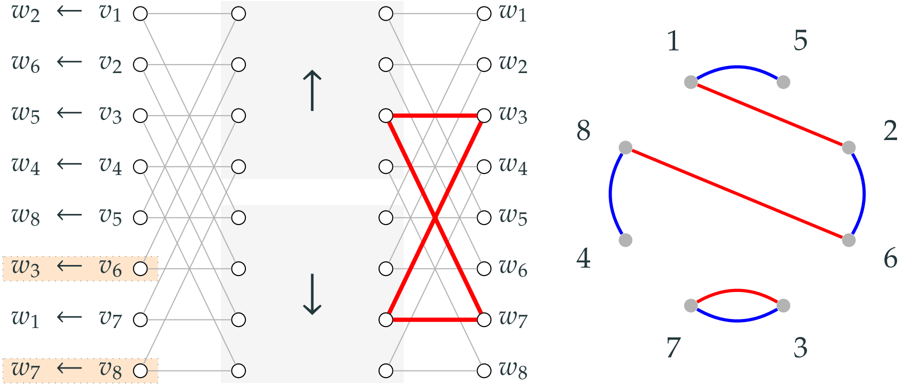
For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.



Theorem

[Beneš'1964]

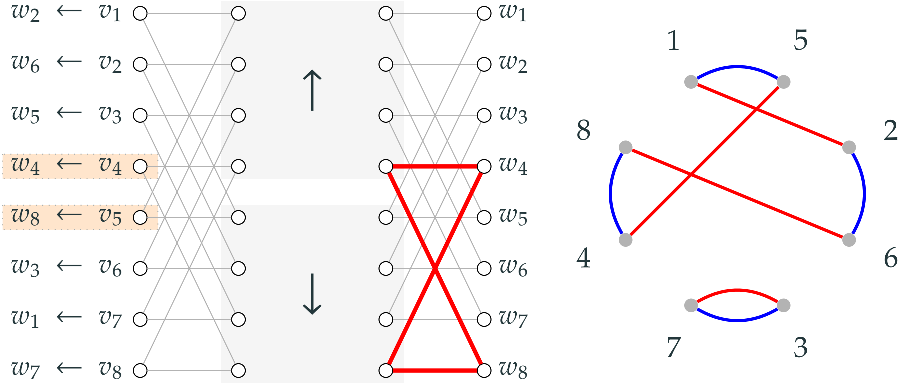
For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.



Theorem

[Beneš'1964]

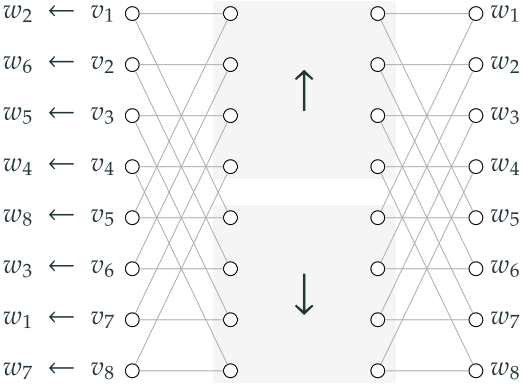
For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.

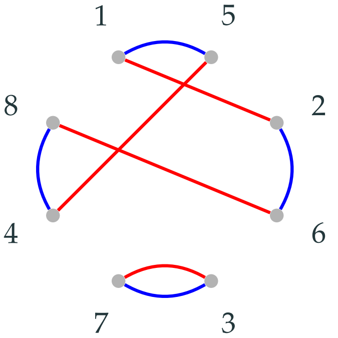


Theorem

[Beneš'1964]

For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.

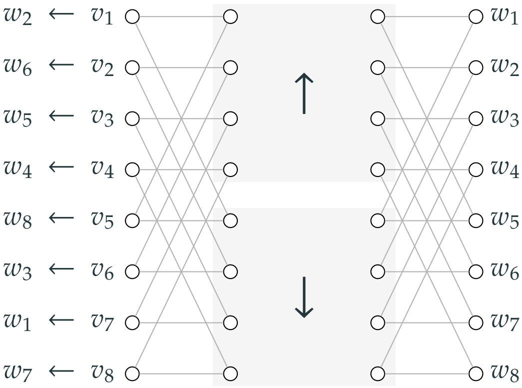


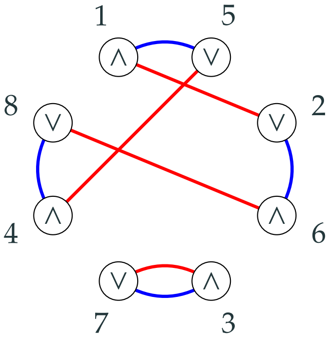


Theorem

[Beneš'1964]

For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.

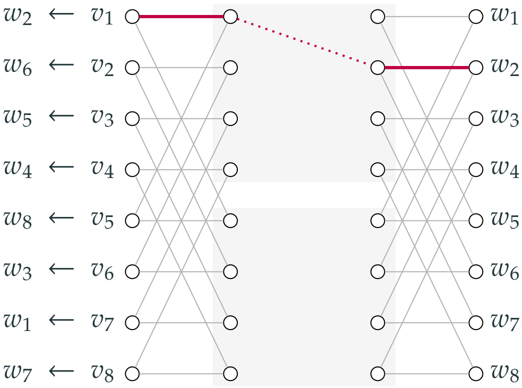


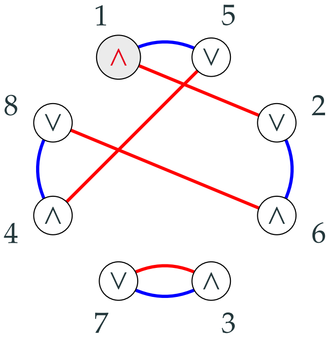


Theorem

[Beneš'1964]

For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.

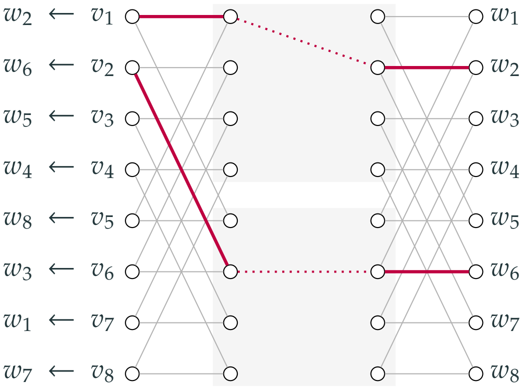


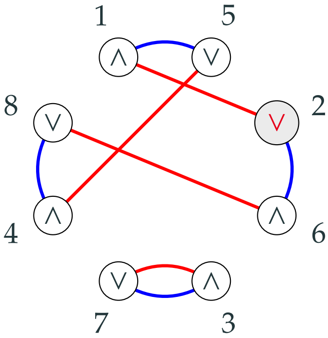


Theorem

[Beneš'1964]

For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.

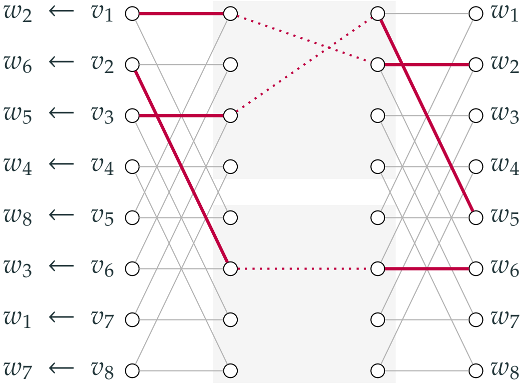


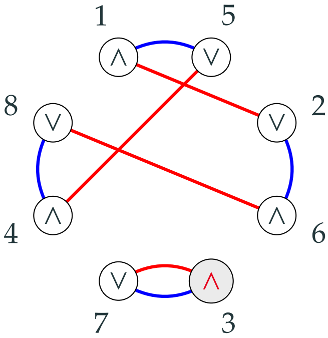


Theorem

[Beneš'1964]

For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.

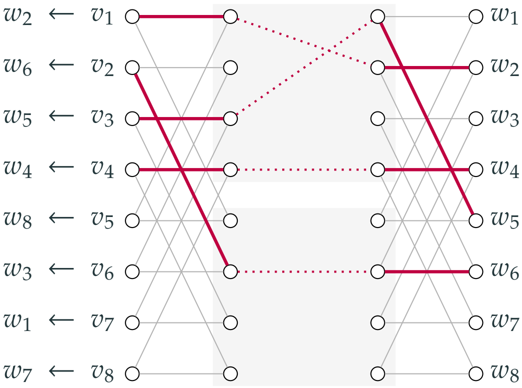


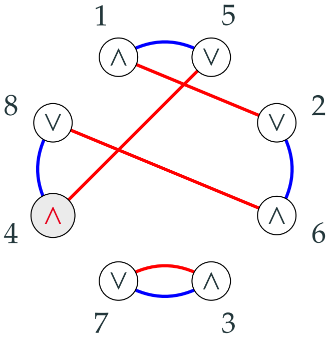


Theorem

[Beneš'1964]

For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.

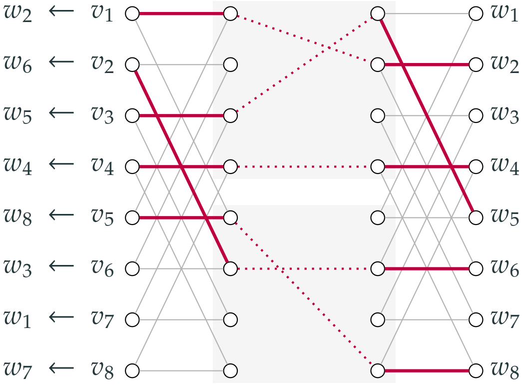


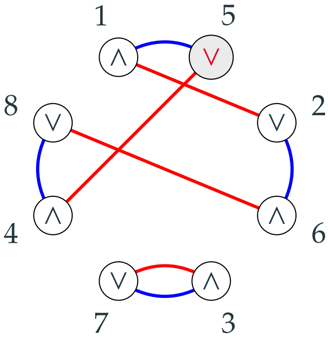


Theorem

[Beneš'1964]

For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.

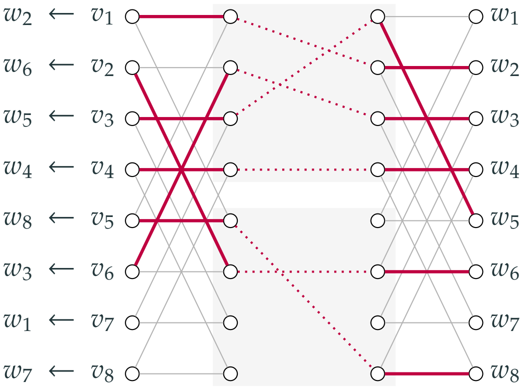


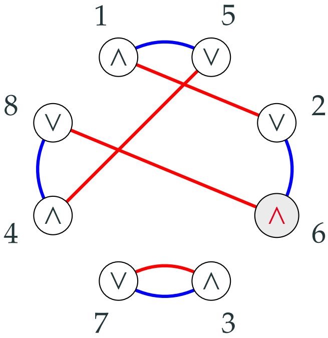


Theorem

[Beneš'1964]

For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.

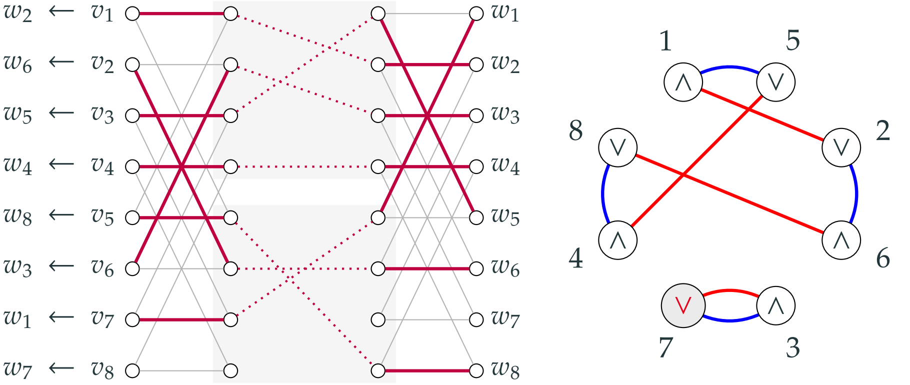




Theorem

[Beneš'1964]

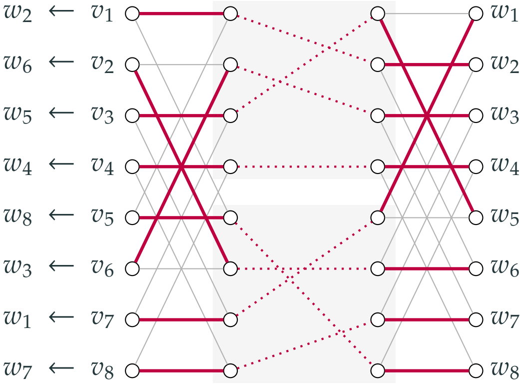
For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.

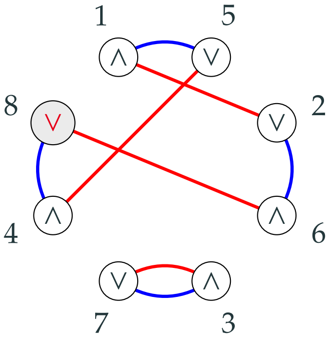


Theorem

[Beneš'1964]

For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.





Theorem

[Beneš'1964]

For any permutation π over $[2^t]$, there is a set of paths $\mathcal{P}$ from v_i to $w_{\pi(i)}$ in B_t such that no vertex appears more than once in $\mathcal{P}$.