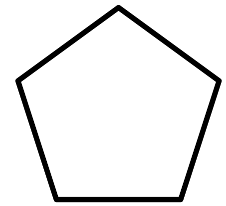
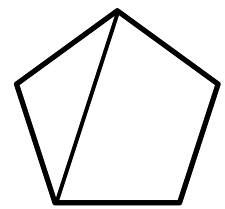
Finding an Easily Recognizable Strong Stable Set or a Meyniel Obstruction in any Graph





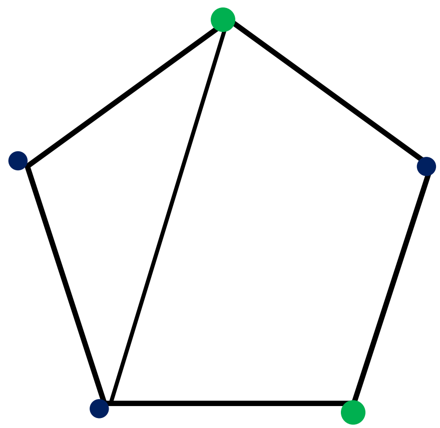
Kathie Cameron

Wilfrid Laurier University

Waterloo, Canada

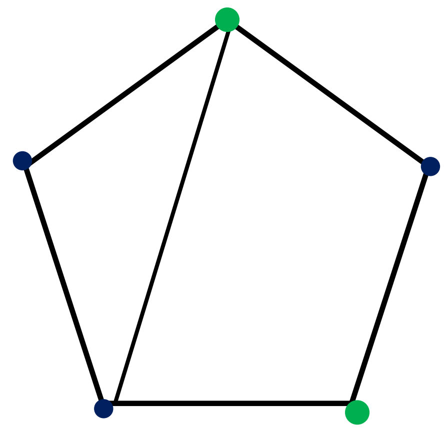
Joint work **with Frédéric Maffray, Jack Edmonds, and Benjamin Lévêque**

A **strong stable set** in G is a stable set which contains a vertex of every maximal clique of G.



Maximal means “with respect to inclusion”

A **strong stable set** in G is a stable set which contains a vertex of every maximal clique of G.



An algorithm for finding a strong stable set can be applied repeatedly to obtain a clique and colouring of the same size.

A **strong stable set** in G is a stable set which contains a vertex of every maximal clique of G.

An algorithm for finding a strong stable set can be applied repeatedly to obtain a clique and colouring of the same size.

A **strong stable set** in G is a stable set which contains a vertex of every maximal clique of G.

An algorithm for finding a strong stable set can be applied repeatedly to obtain a clique and colouring of the same size.

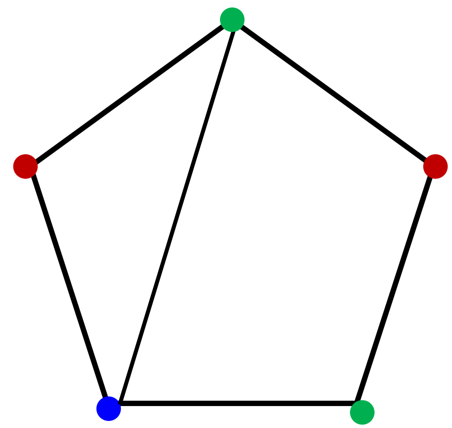
A **strong stable set** in G is a stable set which contains a vertex of every maximal clique of G.

An algorithm for finding a strong stable set can be applied repeatedly to obtain a clique and colouring of the same size.

A **strong stable set** in G is a stable set which contains a vertex of every maximal clique of G.

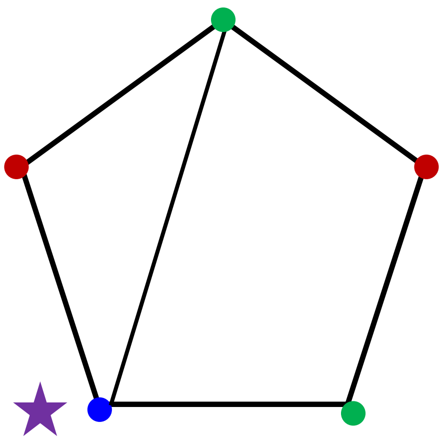
An algorithm for finding a strong stable set can be applied repeatedly to obtain a clique and colouring of the same size.

A **strong stable set** in G is a stable set which contains a vertex of every maximal clique of G.



An algorithm for finding a strong stable set can be applied repeatedly to obtain a clique and colouring of the same size.

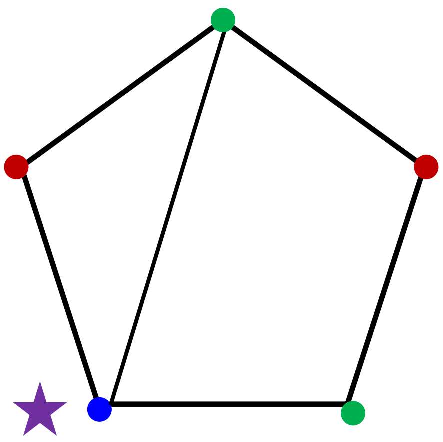
A **strong stable set** in G is a stable set which contains a vertex of every maximal clique of G.



An algorithm for finding a strong stable set can be applied repeatedly to obtain a clique and colouring of the same size.

To find a clique of the same size, start with a vertex of the last colour.

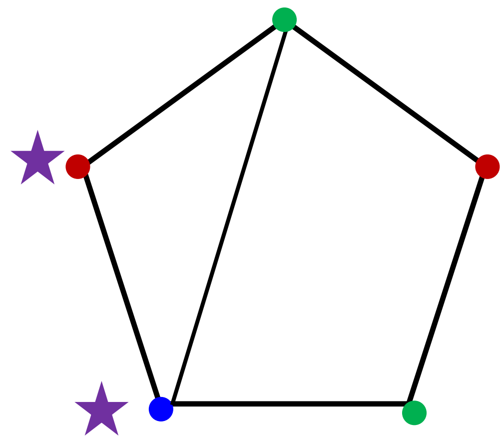
A **strong stable set** in G is a stable set which contains a vertex of every maximal clique of G.



An algorithm for finding a strong stable set can be applied repeatedly to obtain a clique and colouring of the same size.

To find a clique of the same size, start with a vertex of the last colour. It was adjacent to a vertex of the second-last colour.

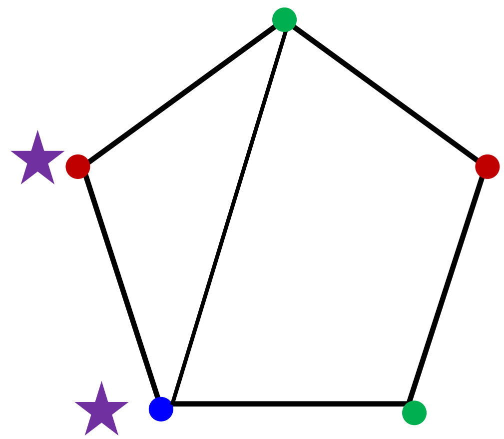
A **strong stable set** in G is a stable set which contains a vertex of every maximal clique of G.



An algorithm for finding a strong stable set can be applied repeatedly to obtain a clique and colouring of the same size.

To find a clique of the same size, start with a vertex of the last colour. It was adjacent to a vertex of the second-last colour.

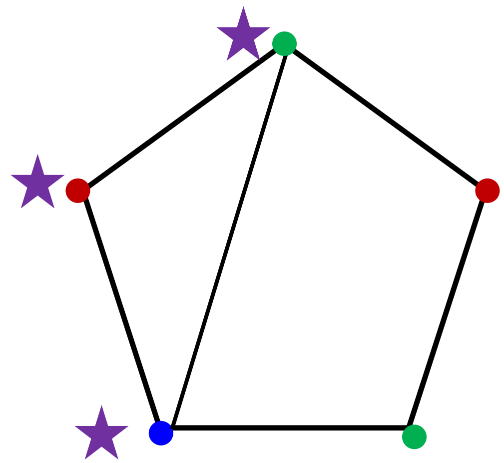
A **strong stable set** in G is a stable set which contains a vertex of every maximal clique of G.



An algorithm for finding a strong stable set can be applied repeatedly to obtain a clique and colouring of the same size.

To find a clique of the same size, start with a vertex of the last colour. It was adjacent to a vertex of the second-last colour. They were both adjacent to a vertex of the third-last colour.

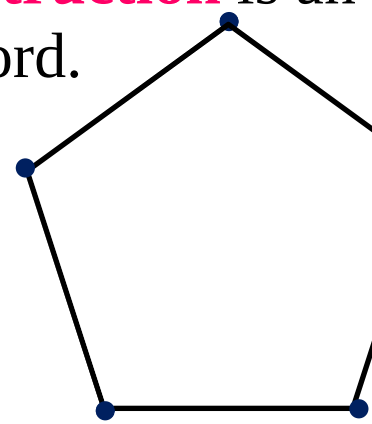
A **strong stable set** in G is a stable set which contains a vertex of every maximal clique of G.

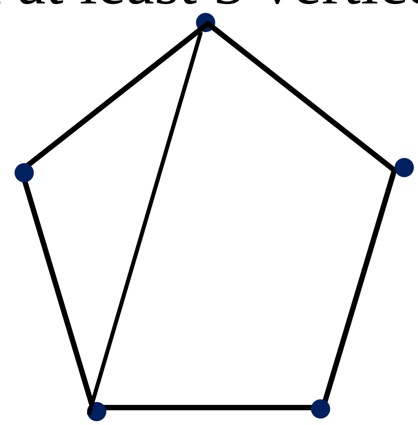


An algorithm for finding a strong stable set can be applied repeatedly to obtain a clique and colouring of the same size.

To find a clique of the same size, start with a vertex of the last colour. It was adjacent to a vertex of the second-last colour. They were both adjacent to a vertex of the third-last colour...

A **Meyniel obstruction** is an odd cycle with at least 5 vertices and at most one chord.





A **Meyniel graph** is a graph with no induced Meyniel obstruction.

Theorem [Meyniel (1976), Markosyan and Karapetyan (1976)]
Meyniel graphs are perfect.

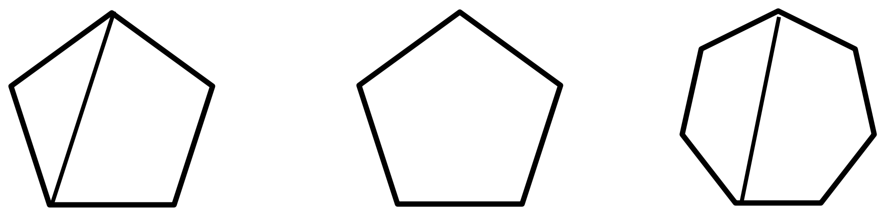
This can be re-stated:
For any graph G,
either G contains a Meyniel obstruction
or G has a clique and colouring of the same size (or both)

Theorem (Hoàng 1987) **Graph G is a Meyniel graph**

if and only if

for every induced subgraph H of G, and every vertex v of H, H contains a strong stable set containing v.

It is easy to see that a Meyniel obstruction contains a vertex which is not in any strong stable set.

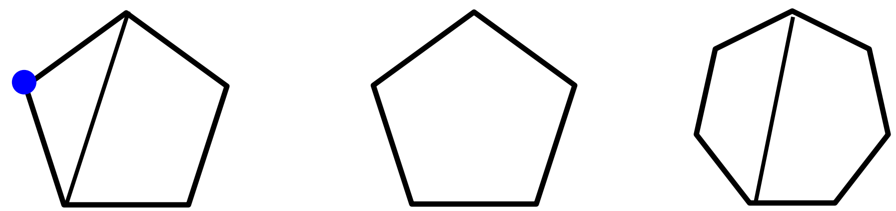


Theorem (Hoàng 1987) **Graph G is a Meyniel graph**

if and only if

for every induced subgraph H of G, and every vertex v of H, H contains a strong stable set containing v.

It is easy to see that a Meyniel obstruction contains a vertex which is not in any strong stable set.



No strong stable set

Thus the main content of Hoàng's Theorem is:

For any graph G and any vertex v of G, either G contains a Meyniel obstruction or G contains a strong stable set containing v (or both).

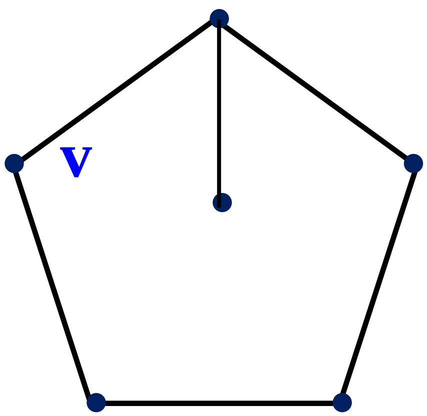
We give a polytime algorithm:

G, v

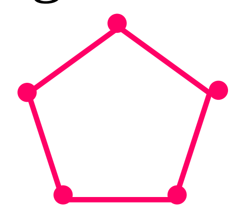
Meyniel obstruction

Strong stable set containing v

If G contains both, we cannot predict which the algorithm will give



Meyniel obstruction



Does not have a strong stable set containing v

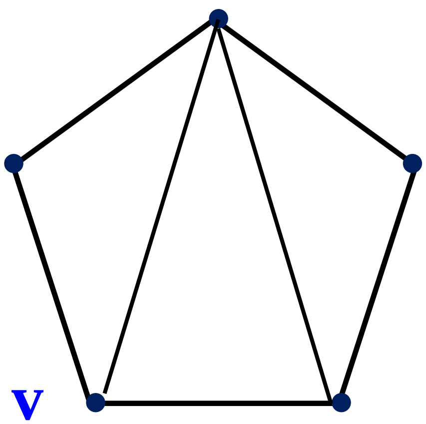
We give a polytime algorithm:

G, v

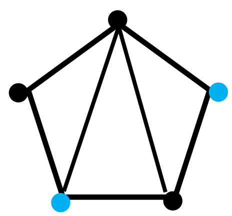
Meyniel obstruction

Strong stable set containing v

If G contains both, we cannot predict which the algorithm will give



Strong stable set containing v



Does not have a Meyniel obstruction

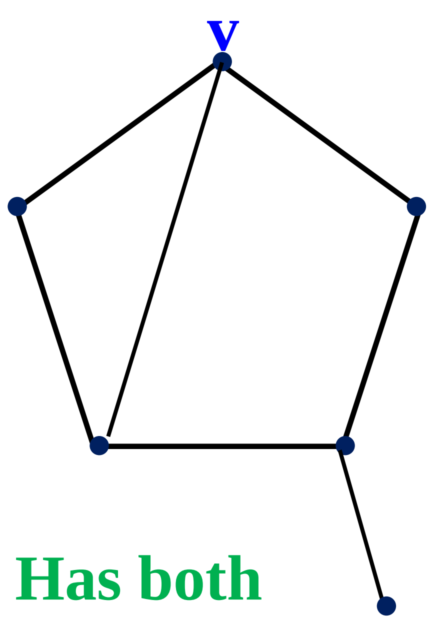
We give a polytime algorithm:

G, v

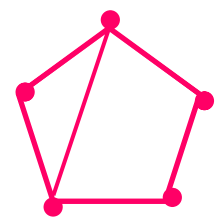
Meyniel obstruction

Strong stable set containing v

If G contains both, we cannot predict which the algorithm will give



Meyniel obstruction



Has both

Algorithm gives one – not both

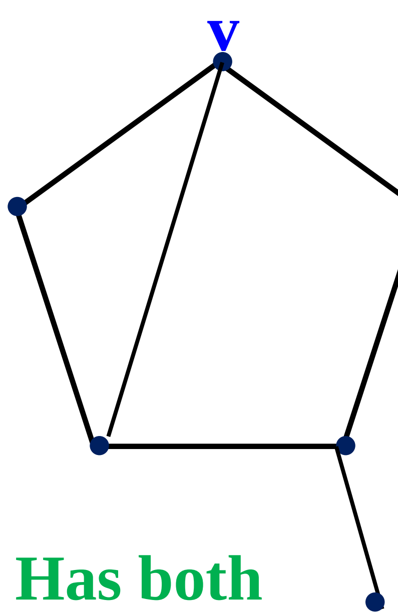
We give a polytime algorithm:

Meyniel obstruction

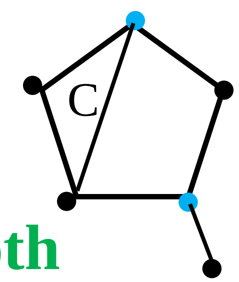
G, v

Strong stable set containing v

If G contains both, we cannot predict which the algorithm will give



strong stable set containing v

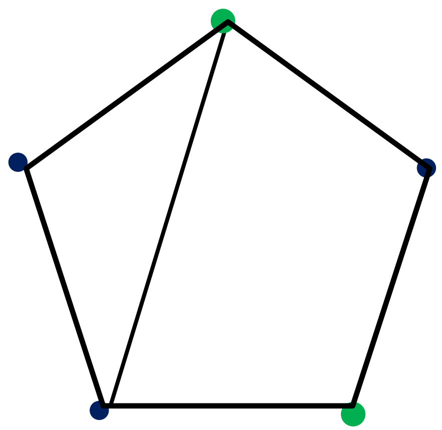


Has both

Algorithm gives one – not both

How can we verify that a stable set is strong?

Recall definition: ***A stable set is strong if it contains a vertex of every maximal clique of G.***

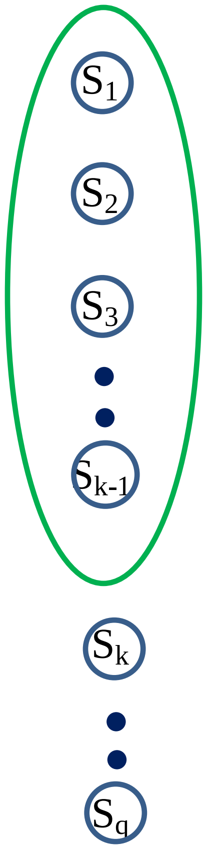


A graph can have an exponential number of maximal cliques.

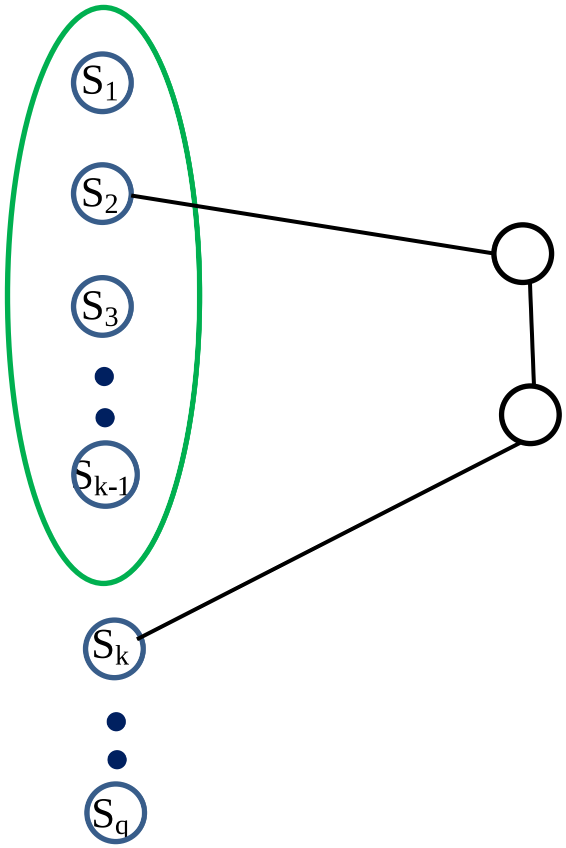
Thus the definition of strong stable set may not be an NP-predicate.

A **nice set** S is a maximal stable set linearly ordered so that there is no induced path on four vertices (P_4) between any vertex u of S and the pseudonode obtained by identifying all vertices of S that precede u.

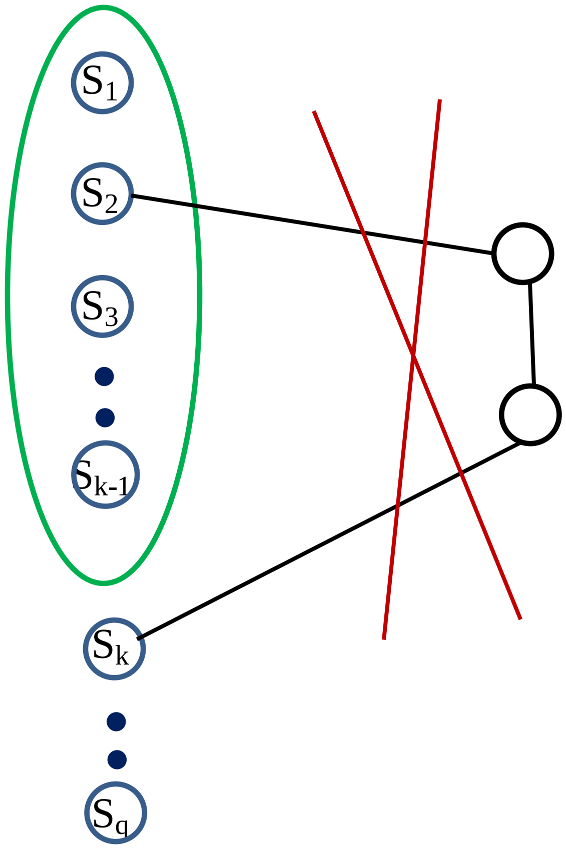
A **nice set** S is a maximal stable set linearly ordered so that there is no induced path on four vertices (P_4) between any vertex u of S and the **pseudonode** obtained by identifying all vertices of S that precede u.



A **nice set** S is a maximal stable set linearly ordered so that there is no induced path on four vertices (P_4) between any vertex u of S and the **pseudonode** obtained by identifying all vertices of S that precede u.



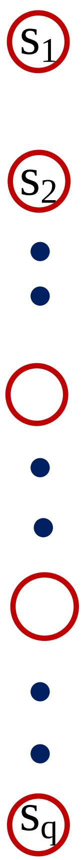
A **nice set** S is a maximal stable set linearly ordered so that there is no induced path on four vertices (P_4) between any vertex u of S and the **pseudonode** obtained by identifying all vertices of S that precede u.



Theorem. **Every nice set is a strong stable set**.

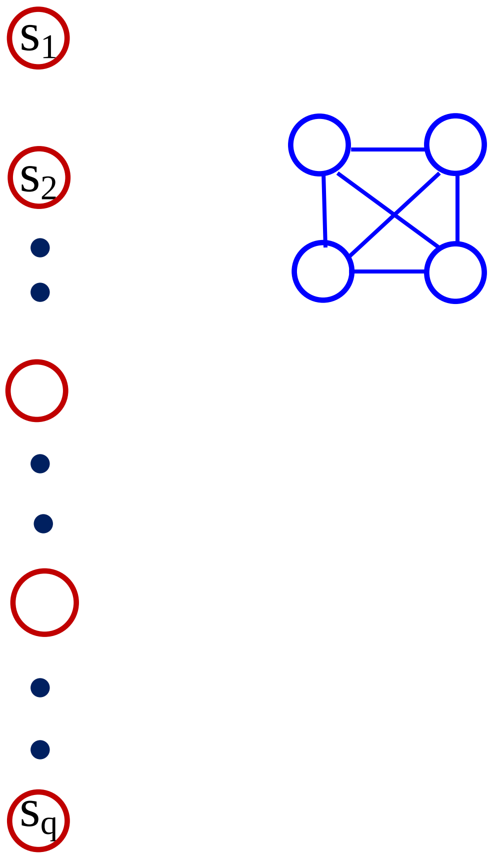
Theorem. **Every nice set is a strong stable set**.

Proof. Let $\mathbf{S = \{s_1, \dots s_q\}}$ be a nice set.



Theorem. **Every nice set is a strong stable set**.

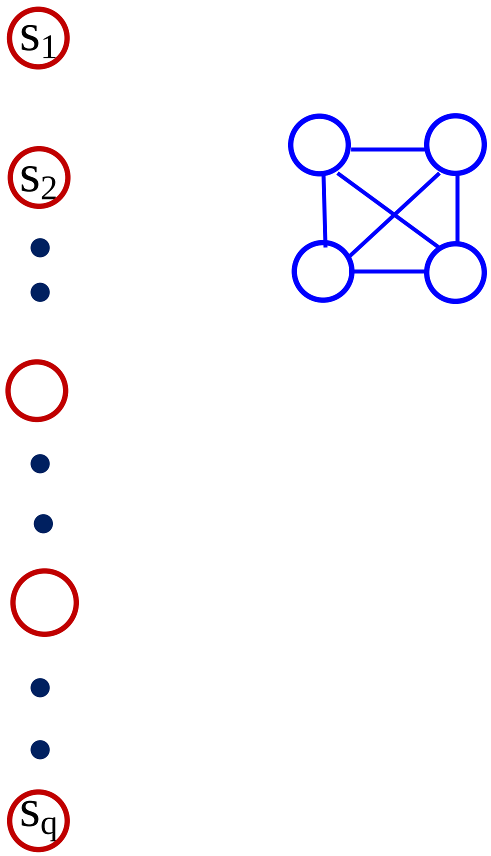
Proof. Let $\mathbf{S = \{s_1, \ldots s_q\}}$ be a nice set. Suppose **C** is a maximal clique in G and that $\mathbf{S} \cap \mathbf{C} = \emptyset$.



Theorem. **Every nice set is a strong stable set**.

Proof. Let $\mathbf{S = \{s_1, \ldots s_q\}}$ be a nice set. Suppose **C** is a maximal clique in G and that $\mathbf{S} \cap \mathbf{C} = \emptyset$.

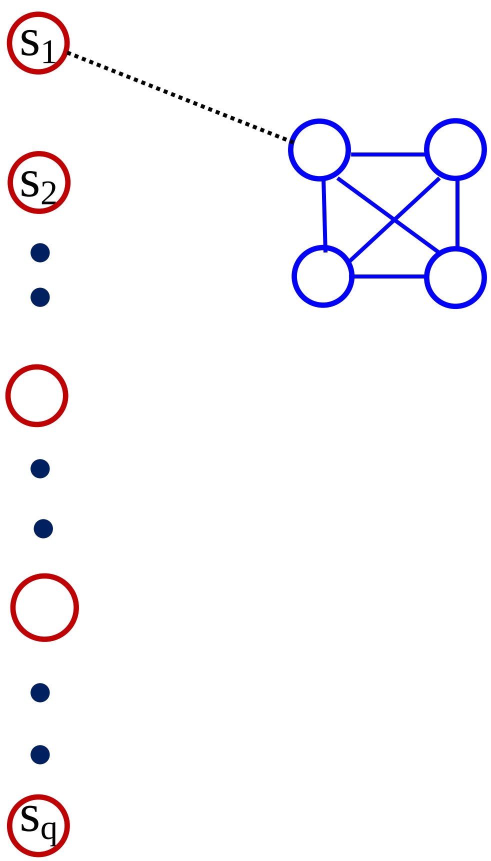
Vertex $\mathbf{s_1}$ can not be adjacent to all of C, since C is a maximal clique.



Theorem. **Every nice set is a strong stable set**.

Proof. Let **$S = \{s_1, \dots s_q\}$** be a nice set. Suppose **C** is a maximal clique in G and that **S** ∩ **C** = ∅.

Vertex **s_1** can not be adjacent to all of C, since C is a maximal clique.

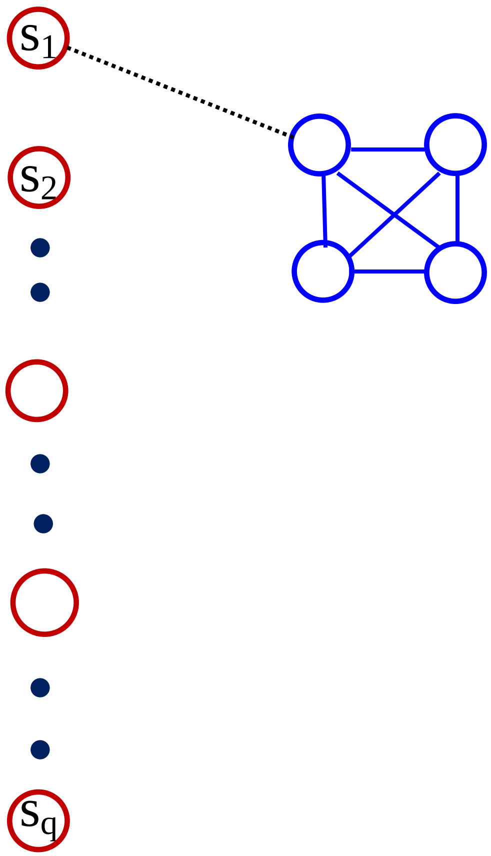


Theorem. **Every nice set is a strong stable set**.

Proof. Let $\mathbf{S = \{s_1, \dots s_q\}}$ be a nice set. Suppose **C** is a maximal clique in G and that $\mathbf{S} \cap \mathbf{C} = \emptyset$.

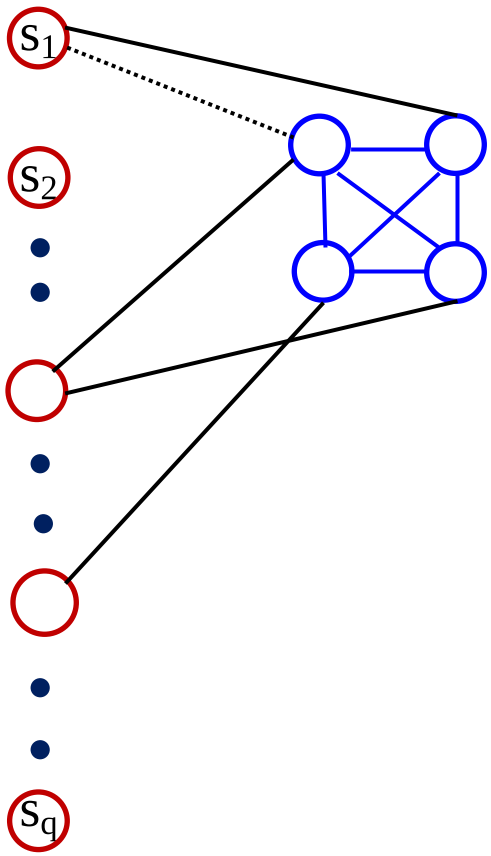
Vertex $\mathbf{s_1}$ can not be adjacent to all of C, since C is a maximal clique.

Since S is a maximal stable set, every vertex of C is adjacent to some vertex of S.



Theorem. **Every nice set is a strong stable set**.

Proof. Let $\mathbf{S = \{s_1, \ldots s_q\}}$ be a nice set. Suppose **C** is a maximal clique in G and that **S** ∩ **C** = ∅.



Vertex $\mathbf{s_1}$ can not be adjacent to all of **C**, since **C** is a maximal clique.

Since **S** is a maximal stable set, every vertex of **C** is adjacent to some vertex of **S**.

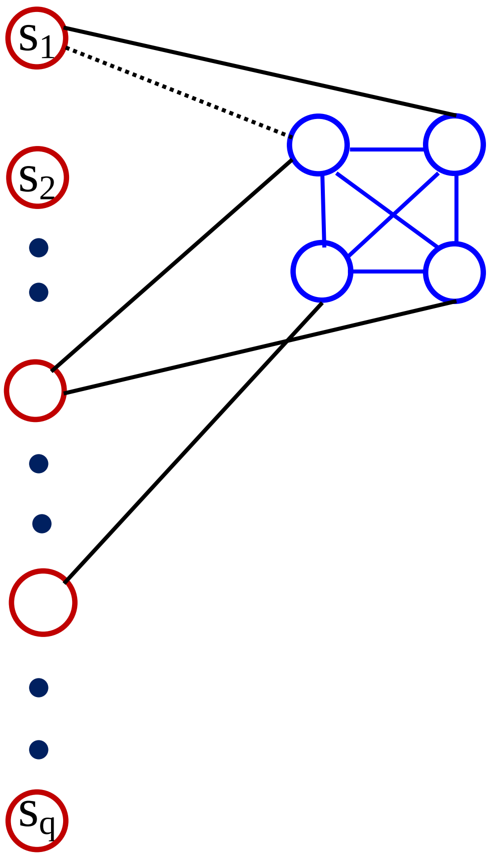
Theorem. **Every nice set is a strong stable set**.

Proof. Let **$S = \{s_1, \ldots s_q\}$** be a nice set. Suppose **C** is a maximal clique in G and that **S** ∩ **C** = ∅.

Vertex **s_1** can not be adjacent to all of **C**, since **C** is a maximal clique.

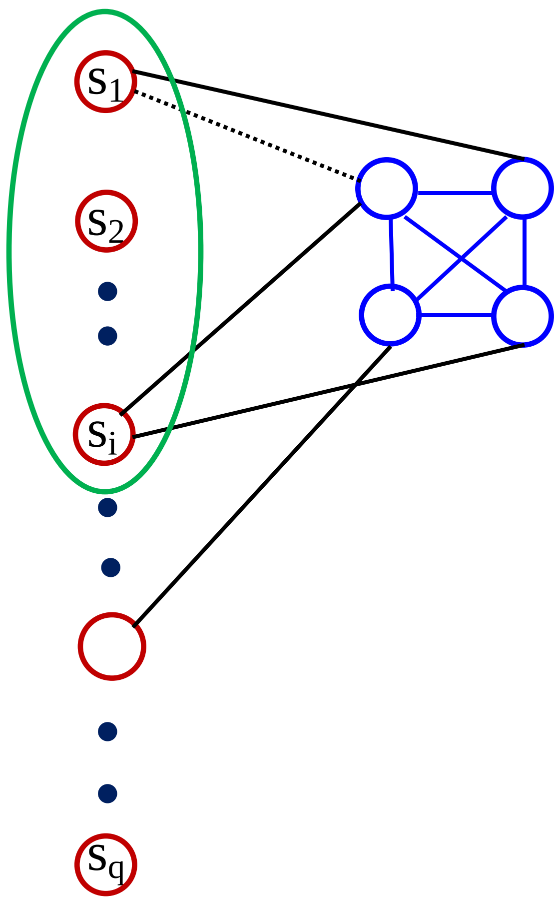
Since **S** is a maximal stable set, every vertex of **C** is adjacent to some vertex of **S**.

Let w_i be the pseudonode consisting of **$\{s_1, \ldots s_i\}$**



Theorem. **Every nice set is a strong stable set**.

Proof. Let $\mathbf{S = \{s_1, \ldots s_q\}}$ be a nice set. Suppose **C** is a maximal clique in G and that **S** ∩ **C** = ∅.



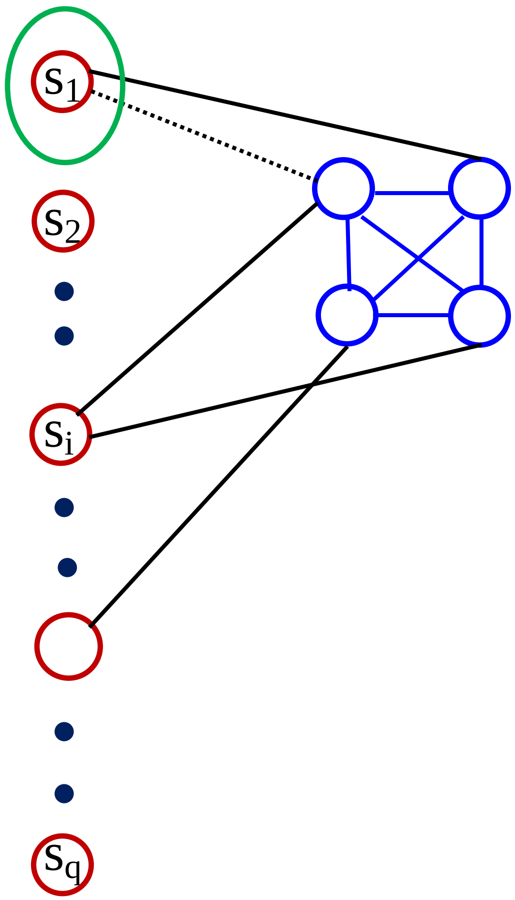
Vertex $\mathbf{s_1}$ can not be adjacent to all of **C**, since **C** is a maximal clique.

Since **S** is a maximal stable set, every vertex of **C** is adjacent to some vertex of **S**.

Let w_i be the **pseudonode** consisting of $\mathbf{\{s_1, \ldots s_i\}}$

Theorem. **Every nice set is a strong stable set**.

Proof. Let $\mathbf{S = \{s_1, \ldots s_q\}}$ be a nice set. Suppose **C** is a maximal clique in G and that **S** $\cap$ **C** $= \emptyset$.



Vertex $\mathbf{s_1}$ can not be adjacent to all of **C**, since **C** is a maximal clique.

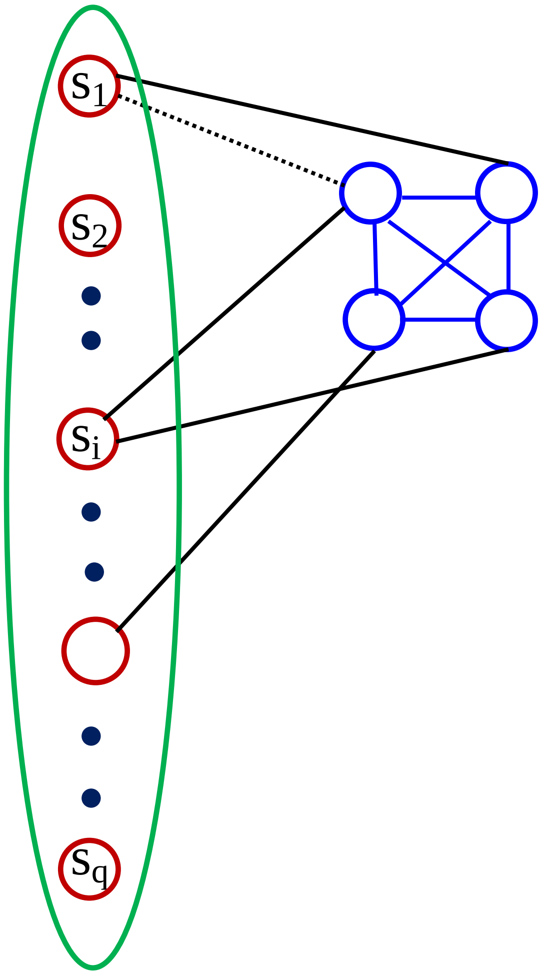
Since **S** is a maximal stable set, every vertex of **C** is adjacent to some vertex of **S**.

Let $\mathbf{w_i}$ be the **pseudonode** consisting of $\mathbf{\{s_1, \ldots s_i\}}$

Pseudonode $\mathbf{w_1}$ is not adjacent to all of **C**

Theorem. **Every nice set is a strong stable set**.

Proof. Let **$S = \{s_1, \ldots s_q\}$** be a nice set. Suppose **C** is a maximal clique in G and that **S** ∩ **C** = ∅.



Vertex **s_1** can not be adjacent to all of **C**, since **C** is a maximal clique.

Since **S** is a maximal stable set, every vertex of **C** is adjacent to some vertex of **S**.

Let **w_i** be the **pseudonode** consisting of **$\{s_1, \ldots s_i\}$**

Pseudonode w_1 is not adjacent to all of **C**

Pseudonode w_q is adjacent to all of **C**

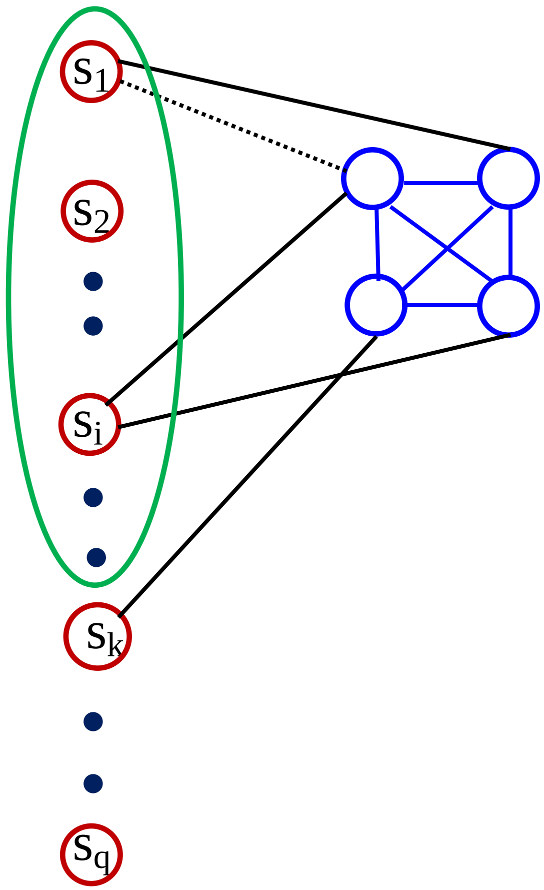
Theorem. **Every nice set is a strong stable set**.

Proof. Let $\mathbf{S = \{s_1, \ldots s_q\}}$ be a nice set. Suppose $\mathbf{C}$ is a maximal clique in G and that $\mathbf{S} \cap \mathbf{C} = \emptyset$.

Vertex $\mathbf{s_1}$ can not be adjacent to all of **C**, since **C** is a maximal clique.

Since **S** is a maximal stable set, every vertex of **C** is adjacent to some vertex of **S**.

Let $\mathbf{w_i}$ be the **pseudonode** consisting of $\mathbf{\{s_1, \ldots s_i\}}$



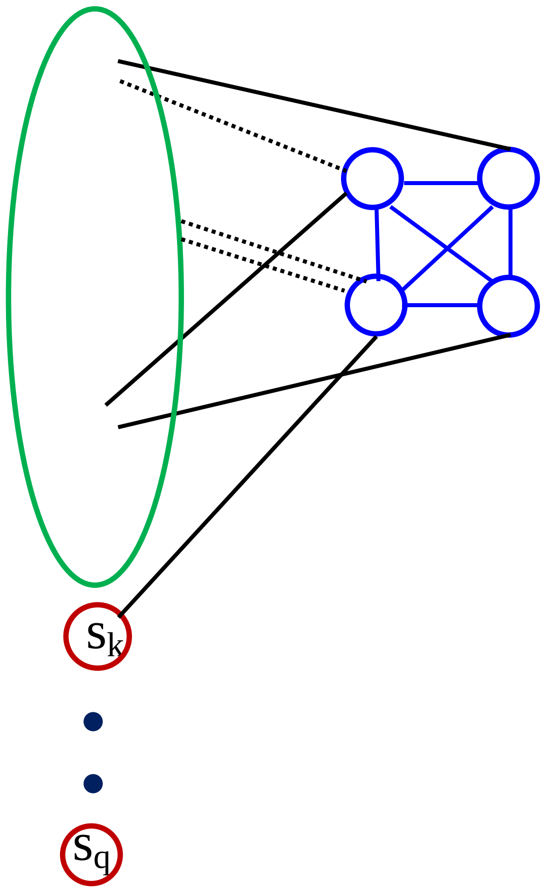
Pseudonode $\mathbf{w_1}$ is not adjacent to all of **C**

Pseudonode $\mathbf{w_q}$ is adjacent to all of **C**

Let $\mathbf{w_{k-1}}$ be the a pseudonode such that $\mathbf{w_{k-1}}$ is not adjacent to all of **C** but $\mathbf{w_k}$ is.

Theorem. **Every nice set is a strong stable set**.

Proof. Let $\mathbf{S = \{s_1, \ldots s_q\}}$ be a nice set. Suppose $\mathbf{C}$ is a maximal clique in G and that $\mathbf{S} \cap \mathbf{C} = \emptyset$.



Vertex $\mathbf{s_1}$ can not be adjacent to all of $\mathbf{C}$, since $\mathbf{C}$ is a maximal clique.

Since $\mathbf{S}$ is a maximal stable set, every vertex of $\mathbf{C}$ is adjacent to some vertex of $\mathbf{S}$.

Let $\mathbf{w_i}$ be the **pseudonode** consisting of $\mathbf{\{s_1, \ldots s_i\}}$

Pseudonode $\mathbf{w_1}$ is not adjacent to all of $\mathbf{C}$

Pseudonode $\mathbf{w_q}$ is adjacent to all of $\mathbf{C}$

Let $\mathbf{w_{k-1}}$ be the a pseudonode such that $\mathbf{w_{k-1}}$ is not adjacent to all of $\mathbf{C}$ but $\mathbf{w_k}$ is.

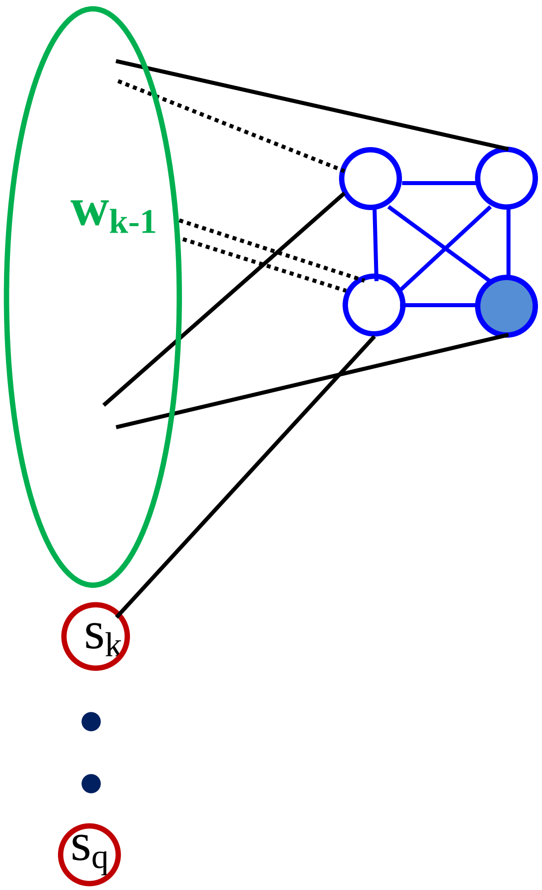
Theorem. **Every nice set is a strong stable set.**

Proof. Let $\mathbf{S = \{s_1, \ldots s_q\}}$ be a nice set. Suppose **C** is a maximal clique in G and that **S** $\cap$ **C** $= \emptyset$.

Vertex $\mathbf{s_1}$ can not be adjacent to all of **C**, since **C** is a maximal clique.

Since **S** is a maximal stable set, every vertex of **C** is adjacent to some vertex of **S**.

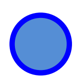
Let $\mathbf{w_i}$ be the **pseudonode** consisting of $\mathbf{\{s_1, \ldots s_i\}}$



Pseudonode $\mathbf{w_1}$ **is not** adjacent to all of **C**

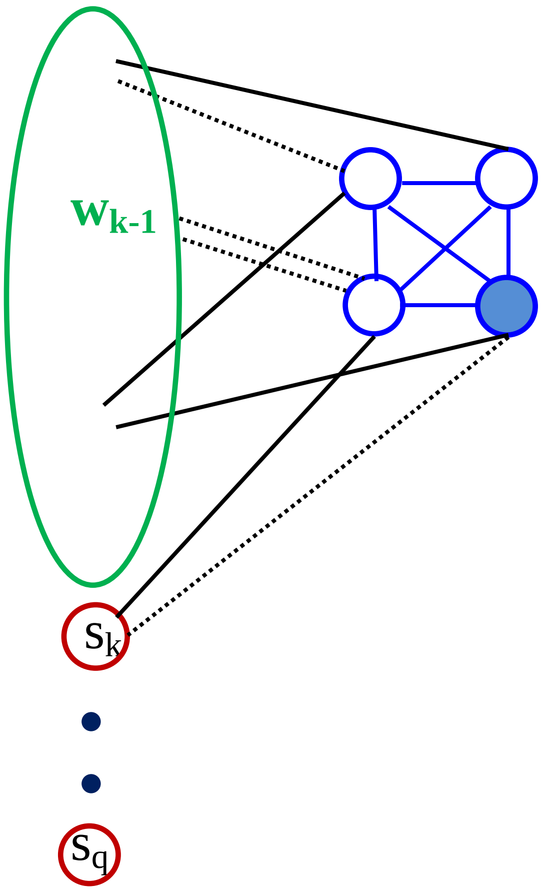
Pseudonode $\mathbf{w_q}$ **is** adjacent to all of **C**

Let $\mathbf{w_{k-1}}$ be the a pseudonode such that $\mathbf{w_{k-1}}$ is not adjacent to all of **C** but $\mathbf{w_k}$ is.

Let  be a vertex of **C** which is not adjacent to $\mathbf{s_k}$

Theorem. **Every nice set is a strong stable set**.

Proof. Let $\mathbf{S = \{s_1, \ldots s_q\}}$ be a nice set. Suppose **C** is a maximal clique in G and that **S** $\cap$ **C** $= \emptyset$.



Vertex $\mathbf{s_1}$ can not be adjacent to all of **C**, since **C** is a maximal clique.

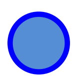
Since **S** is a maximal stable set, every vertex of **C** is adjacent to some vertex of **S**.

Let $\mathbf{w_i}$ be the **pseudonode** consisting of $\mathbf{\{s_1, \ldots s_i\}}$

Pseudonode $\mathbf{w_1}$ is not adjacent to all of **C**

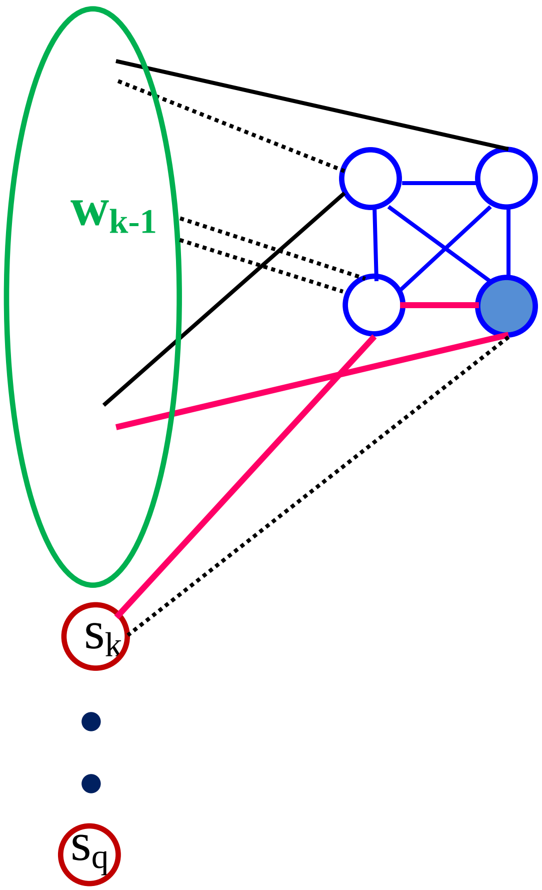
Pseudonode $\mathbf{w_q}$ is adjacent to all of **C**

Let $\mathbf{w_{k-1}}$ be the a pseudonode such that $\mathbf{w_{k-1}}$ is not adjacent to all of **C** but $\mathbf{w_k}$ is.

Let  be a vertex of **C** which is not adjacent to $\mathbf{s_k}$

Theorem. **Every nice set is a strong stable set**.

Proof. Let $\mathbf{S = \{s_1, \ldots s_q\}}$ be a nice set. Suppose **C** is a maximal clique in G and that **S** ∩ **C** = ∅.



Vertex $\mathbf{s_1}$ can not be adjacent to all of **C**, since **C** is a maximal clique.

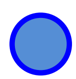
Since **S** is a maximal stable set, every vertex of **C** is adjacent to some vertex of **S**.

Let $\mathbf{w_i}$ be the **pseudonode** consisting of $\mathbf{\{s_1, \ldots s_i\}}$

Pseudonode $\mathbf{w_1}$ is not adjacent to all of **C**

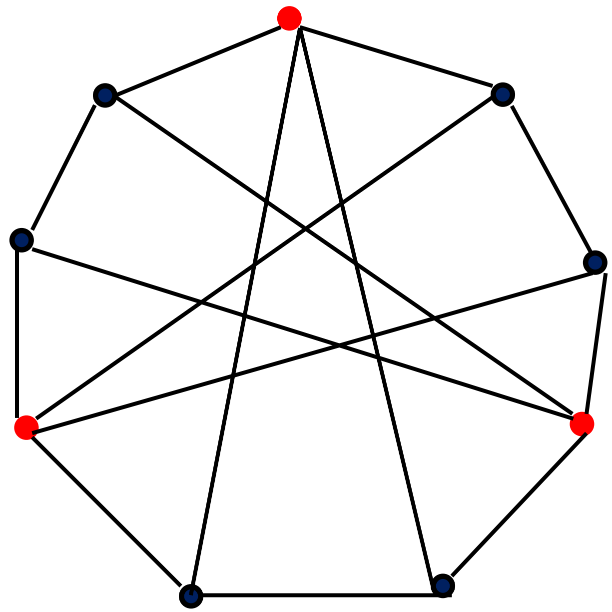
Pseudonode $\mathbf{w_q}$ is adjacent to all of **C**

Let $\mathbf{w_{k-1}}$ be the a pseudonode such that $\mathbf{w_{k-1}}$ is not adjacent to all of **C** but $\mathbf{w_k}$ is.

Let  be a vertex of **C** which is not adjacent to $\mathbf{s_k}$

There is a $\mathbf{P_4}$ from **pseudonode $\mathbf{w_{k-1}}$** to $\mathbf{s_k}$ **=> <=**

Not every strong stable set is a nice set.



Recall: Hoàng’s Theorem:

For any graph G and any vertex v of G,
either G contains a Meyniel obstruction or
G contains a <u>strong stable set</u> containing v (or both).

Our algorithm provides the following EP strengthening of Hoàng’s Theorem:

For any graph G and any vertex v of G,
either G contains a Meyniel obstruction or
G contains a <u>nice set</u> containing v (or both).

Previous Work

n = # vertices

m = # edges

Promise G is Meyniel $\Rightarrow$ $\boxed{\text{Polytime Algorithm}}$ $\Rightarrow$ Promise S is a Strong Stable Set

Hoàng (1987) **O(n^7)**

Previous Work

n = # vertices

m = # edges

Promise G is Meyniel ⇒ Polytime Algorithm ⇒ Promise S is a Strong Stable Set

Hoàng (1987) **$O(n^7)$**

KC & Edmonds (2005) **$O(n+m)$**

Better

G, v ⇒ Polytime Algorithm ⇒ Meyniel Obstruction

Polytime Algorithm ⇒ Nice Set S containing vertex v

KC, Lévêque, Maffray (2012) **$O(n^3)$**

KC &Edmonds (2005) **$O(n^2)$**

Algorithm 1

Input: **Graph G and vertex v of G.**
Output: **Nice set containing v or Meyniel obstruction.**

* Let $v = u_1$
* Suppose $u_1, u_2, ..., u_k$, have been chosen.
  - If every vertex of $V(G) - \{ u_1, u_2, ..., u_k\}$ is adjacent to one of $u_1, u_2, ..., u_k$, then the chosen vertices form a nice set.
  - Otherwise, choose u_{k+1} not adjacent to any chosen vertices such that it has the largest nunber of common neighbours with the pseudonode $v(u_1, u_2, ..., u_k)$ obtained by identifying $u_1, u_2, ..., u_k$.
    - If there is a P_4 from $v(u_1, u_2, ..., u_k)$ to u_{k+1}, then G contains a Meyniel obstruction, which we can find using Algorithm 2.
    - Otherwise continue.

Three Levels of Algorithmic Approach

(1) If the input graph is **guaranteed to be Meyniel**, we can omit the step of looking for a P_4 - such a path never exists.

Promise G is Meyniel ⇒ $\boxed{\text{Linear Algorithm}}$ ⇒ Nice Set Containing v

(2) To have a **robust algorithm** in the sense of **Sprinrad**, we can stop as soon as we find a P_4 from $v(u_1, u_2, \ldots, u_k)$ to u_{k+1}, since this indicates that G contains a Meyniel obstruction.

G, v ⇒ $\boxed{\text{Polytime Algorithm}}$ ⇒ **Declare G is not Meyniel**

G, v ⇒ $\boxed{\text{Polytime Algorithm}}$ ⇒ Nice Set Containing v

(3) Algorithm 1 as described is an **EP search algorithm**.

G, v ⇒ $\boxed{\text{Polytime Algorithm}}$ ⇒ **Meyniel Obstruction**

G, v ⇒ $\boxed{\text{Polytime Algorithm}}$ ⇒ **Nice Set Containing v**

Three Levels of Algorithmic Approach

(1) If the input graph is **guaranteed to be Meyniel**, we can omit the step of looking for a P_4 - such a path never exists.

Promise G is Meyniel ⇒ **Linear Algorithm** ⇒ Nice Set Containing v

(2) To have a **robust algorithm** in the sense of **Sprinrad**, we can stop as soon as we find a P_4 from $v(u_1, u_2, \ldots, u_k)$ to u_{k+1}, since this indicates that G contains a Meyniel obstruction.

G, v ⇒ Polytime Algorithm ⇒ **Declare G is not Meyniel** / Nice Set Containing v

(3) Algorithm 1 as described is an **EP search algorithm**.

G, v ⇒ Polytime Algorithm ⇒ **Meyniel Obstruction** / **Nice Set Containing v**

Easily recognizable

Three Levels of Algorithmic Approach

(1) If the input graph is **guaranteed to be Meyniel**, we can omit the step of looking for a P_4 - such a path never exists.

Promise G is Meyniel $\Rightarrow$ $\boxed{\text{Linear Algorithm}}$ $\Rightarrow$ Nice Set Containing v

(2) To have a **robust algorithm** in the sense of **Sprinrad**, we can stop as soon as we find a P_4 from $v(u_1, u_2, \ldots, u_k)$ to u_{k+1}, since this indicates that G contains a Meyniel obstruction.

G, v $\Rightarrow$ $\boxed{\text{Polytime Algorithm}}$

Declare G is not Meyniel

Nice Set Containing v

(3) Algorithm 1 as described is an **EP search algorithm**.

G, v $\Rightarrow$ $\boxed{\text{Polytime Algorithm}}$

Meyniel Obstruction

Nice Set Containing v

Easily recognizable

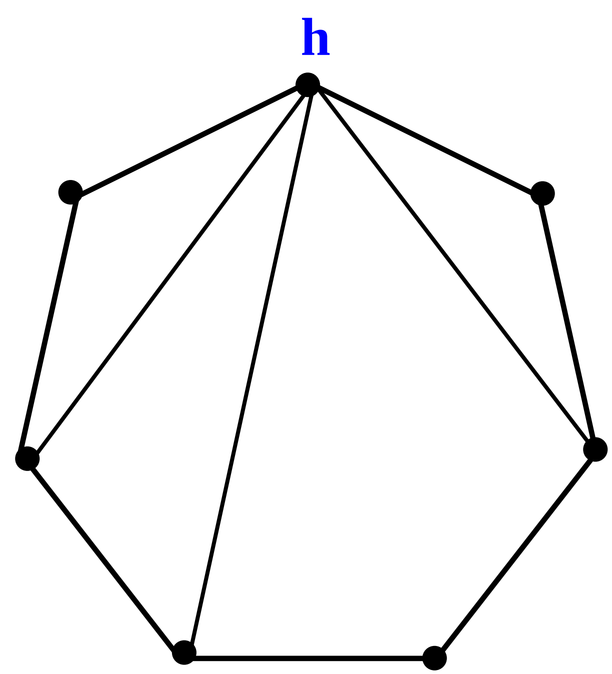
i.e. in NP

Finding a Meyniel Obstruction

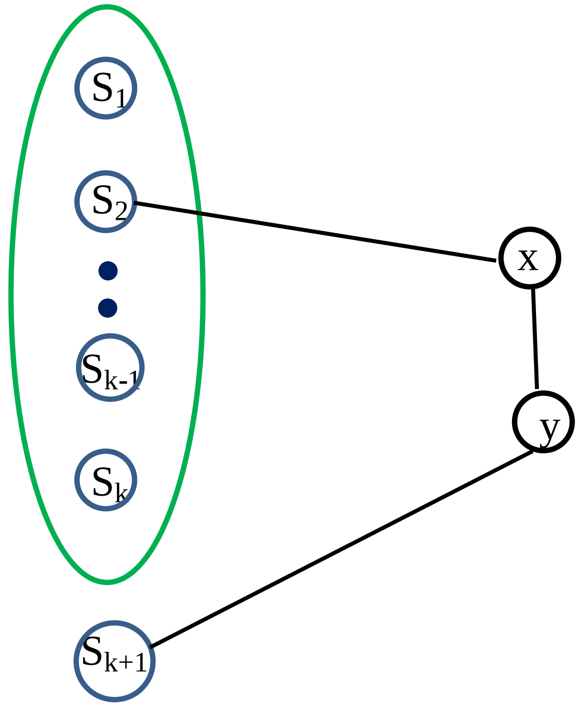
Algorithm says: Choose u_{k+1} not adjacent to any of $u_1, u_2, \ldots, u_k$ such that it has the largest nunber of common neighbours with with the pseudonode $v(u_1, u_2, \ldots, u_k)$ obtained by identifying $u_1, u_2, \ldots, u_k$.

If there is a P_4 from $v(u_1, u_2, \ldots, u_k)$ to u_{k+1}, then G has a Meyniel obstruction.

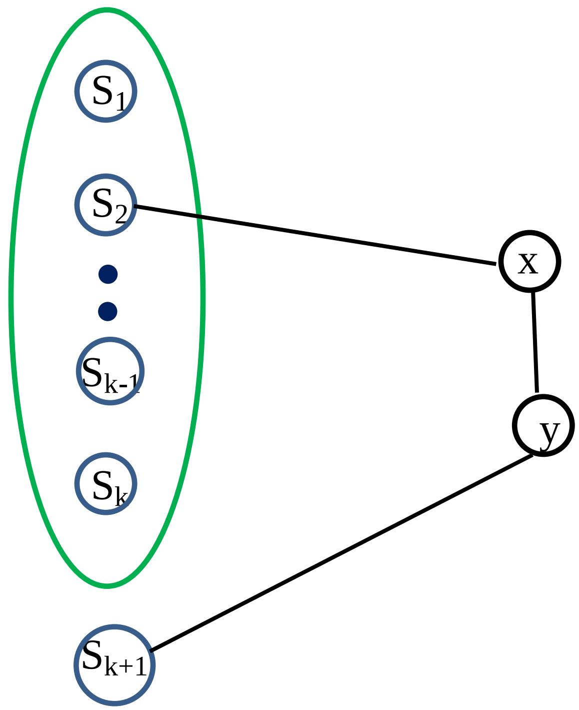
Ravindra's Lemma (1984). In an odd cycle of size at least 5 with all chords hitting the same vertex **h** and at least one of these possible chords missing, there is a Meyniel obstruction
and if the Meyniel obstruction is an odd cycle with one chord, the chord is short and hits **h**.



Pseudonode Expansion

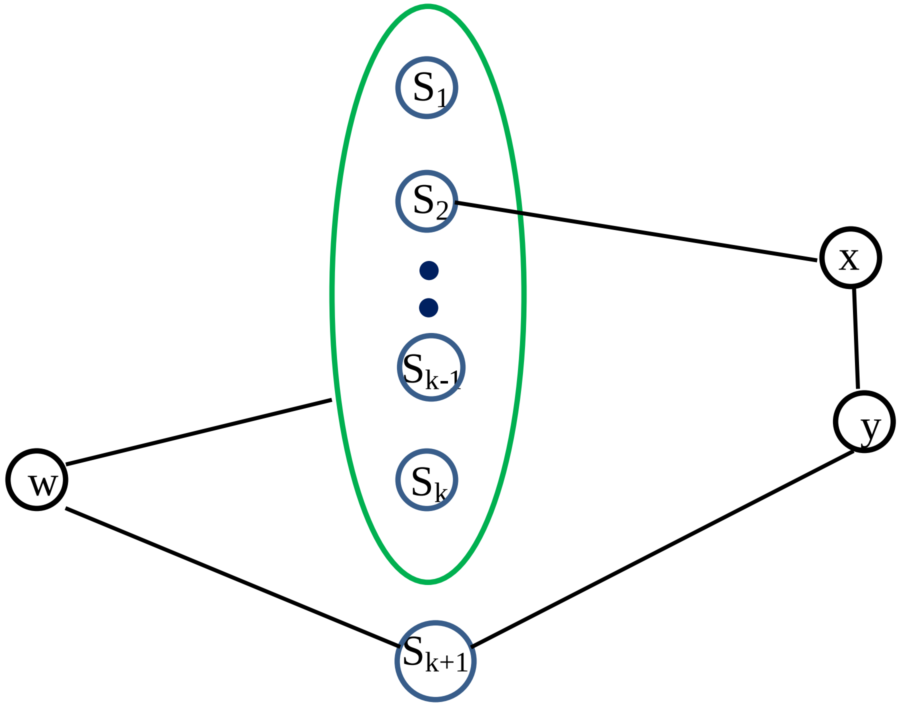


Pseudonode Expansion



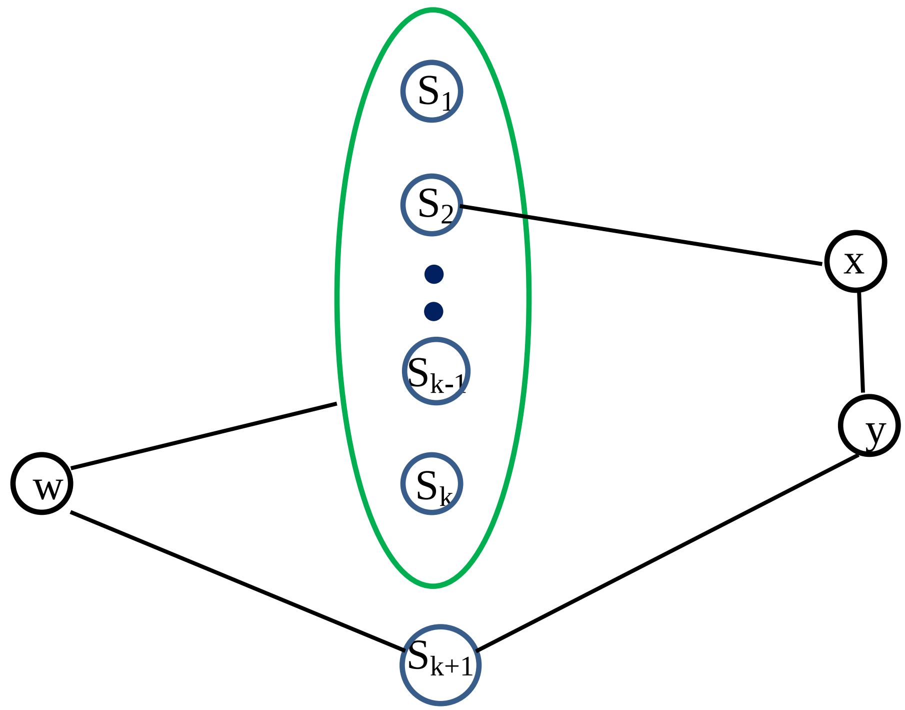
y is non-adjacent to the pseudonode and has common neighbour x with the pseudonode

Pseudonode Expansion



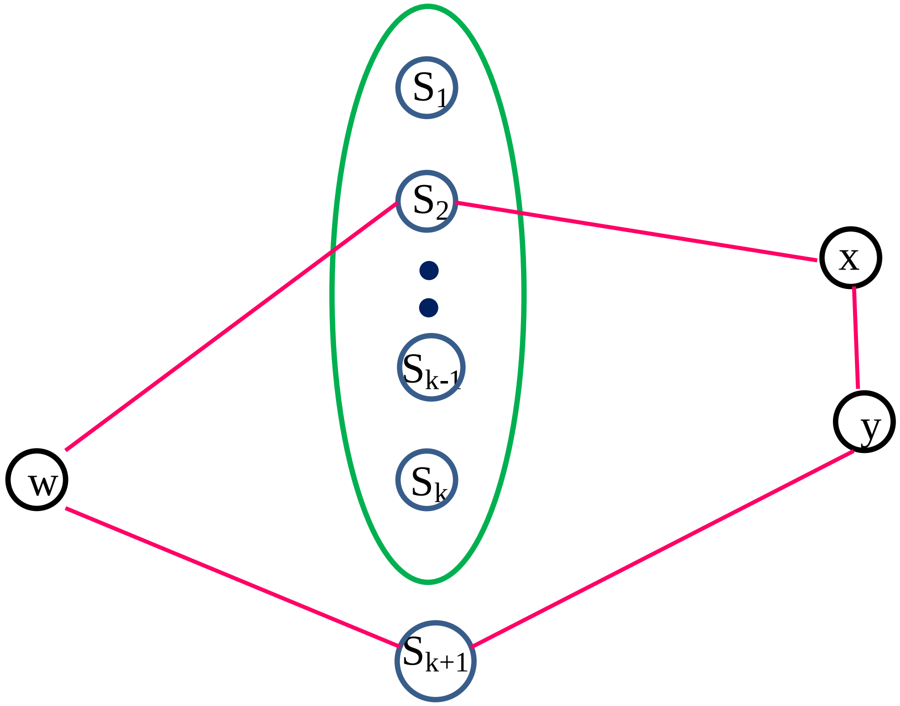
y is non-adjacent to the pseudonode and has common neighbour x with the pseudonode. By choice of s_{k+1} , w exists

Pseudonode Expansion



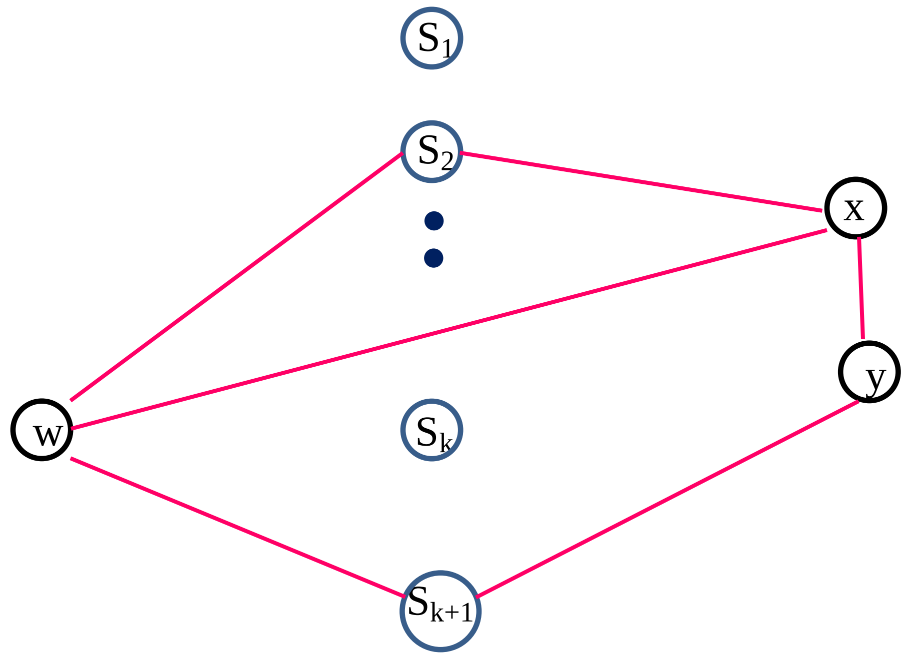
If w and x have a common neighbour in the pseudonode, we have a Meyniel obstruction

Pseudonode Expansion



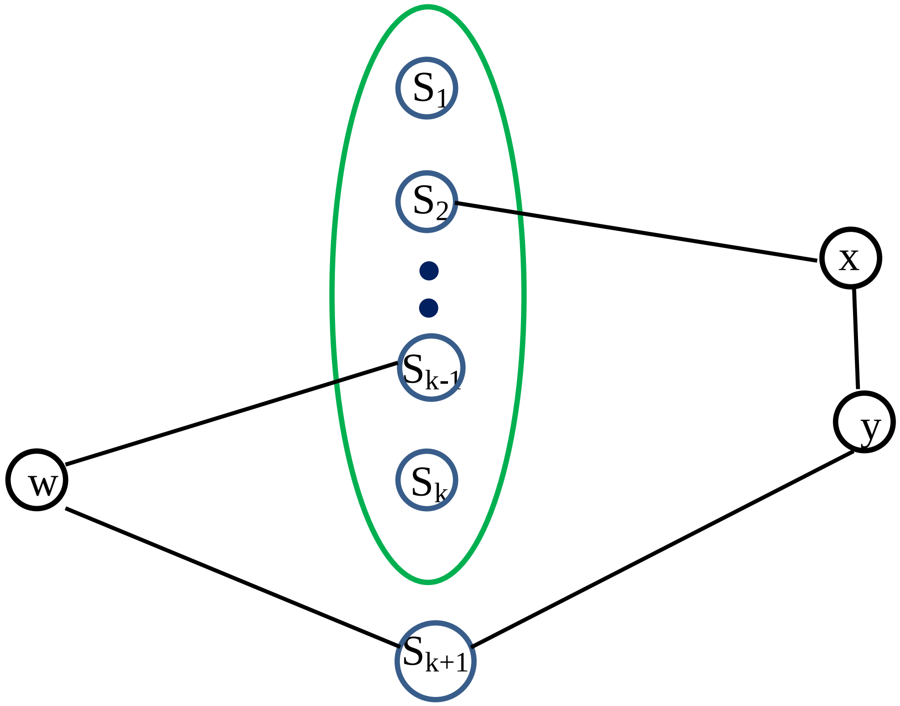
If w and x have a common neighbour in the pseudonode, we have a Meyniel obstruction

Pseudonode Expansion



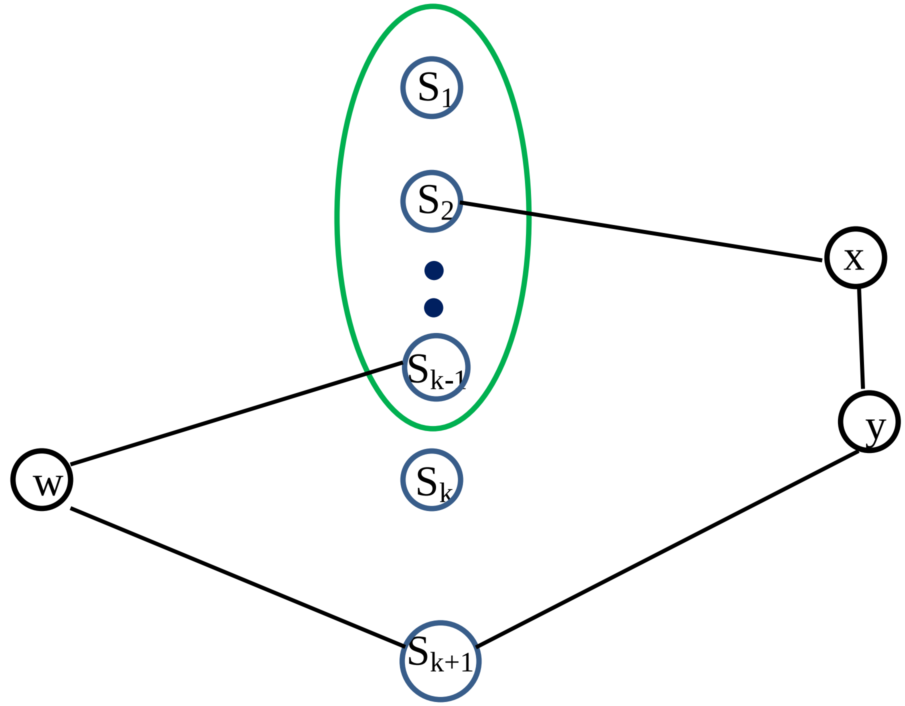
If w and x have a common neighbour in the pseudonode, we have a Meyniel obstruction

Pseudonode Expansion



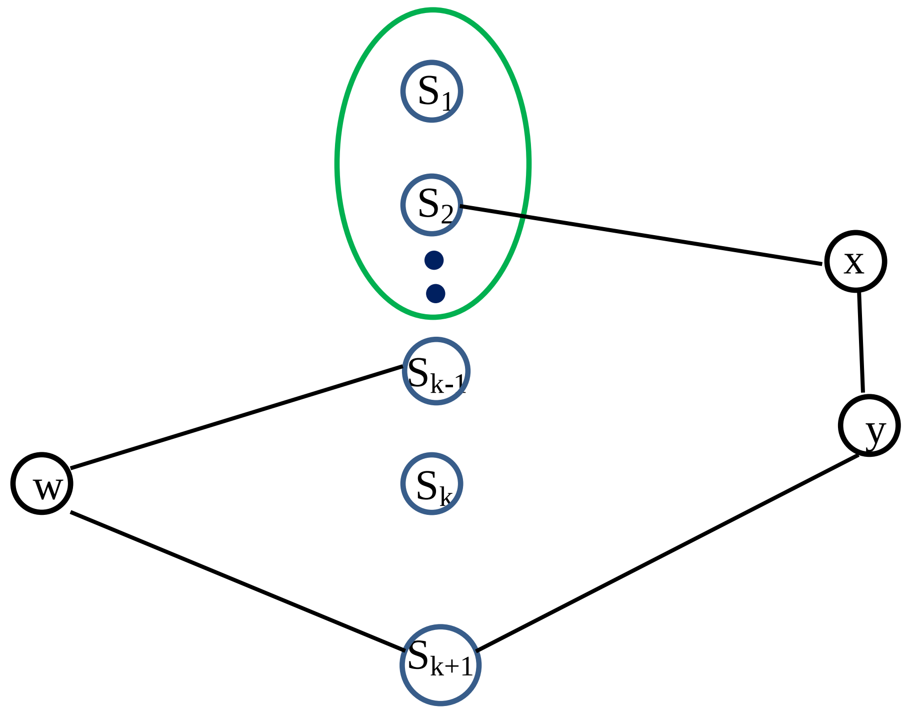
Otherwise, w and x do not have a common neighbour in the pseudonode, and we remove vertices from the pseudonode until the cycle becomes a path

Pseudonode Expansion



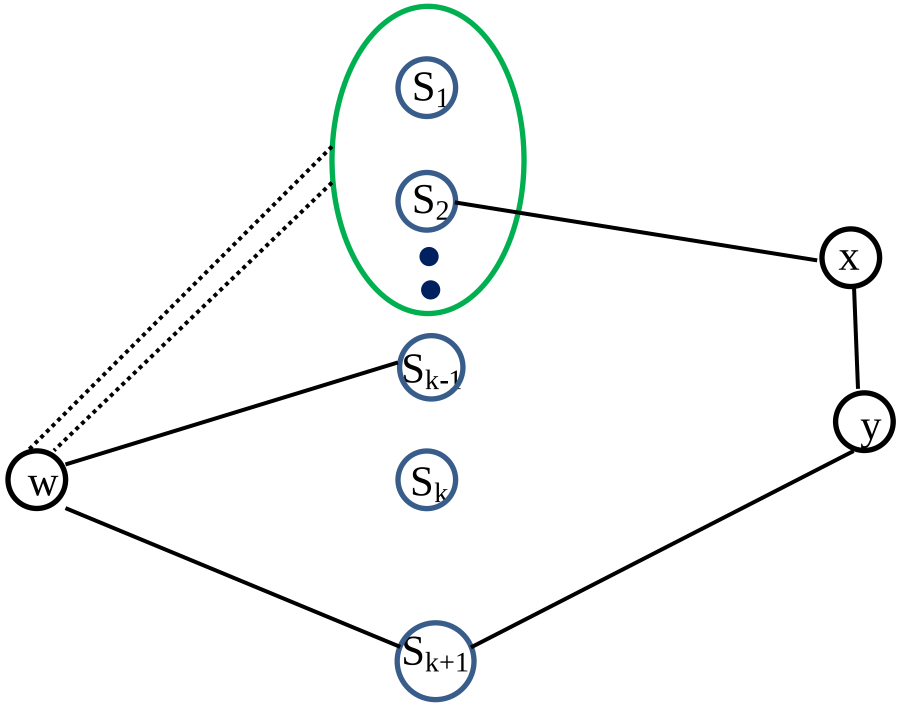
Otherwise, w and x do not have a common neighbour in the pseudonode, and we remove vertices from the pseudonode until the cycle becomes a path

Pseudonode Expansion



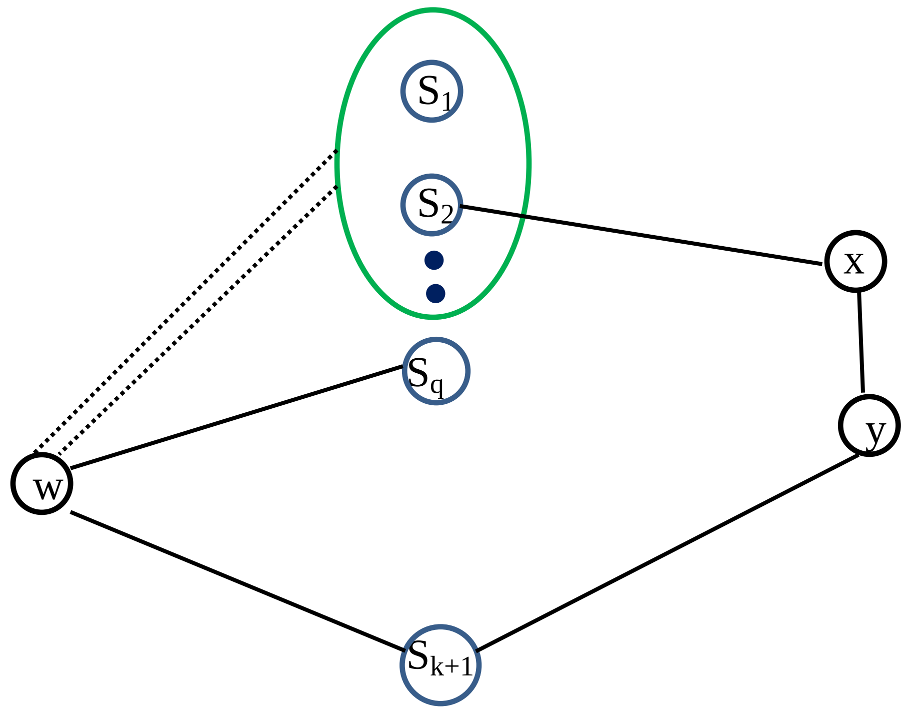
Otherwise, w and x do not have a common neighbour in the pseudonode, and we remove vertices from the pseudonode until the cycle becomes a path

Pseudonode Expansion

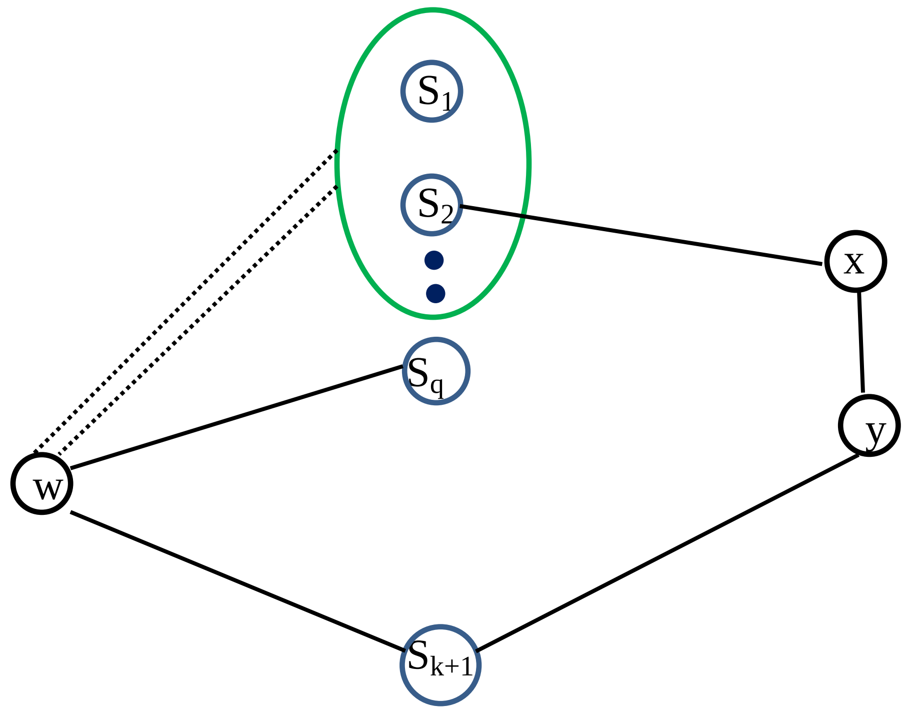


Otherwise, w and x do not have a common neighbour in the pseudonode, and we remove vertices from the pseudonode until the cycle becomes a path

Pseudonode Expansion

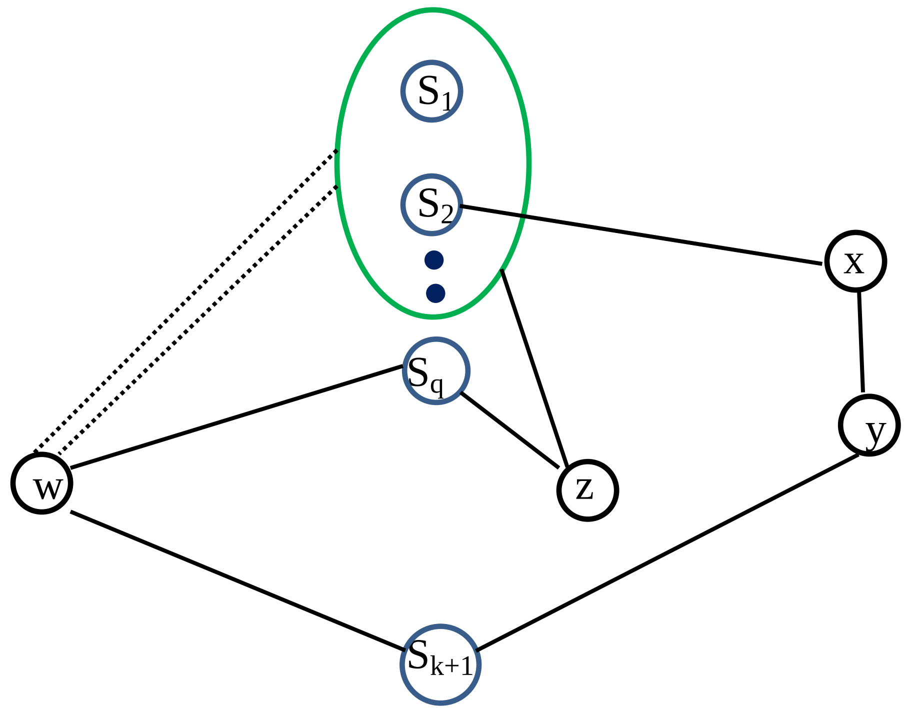


Pseudonode Expansion



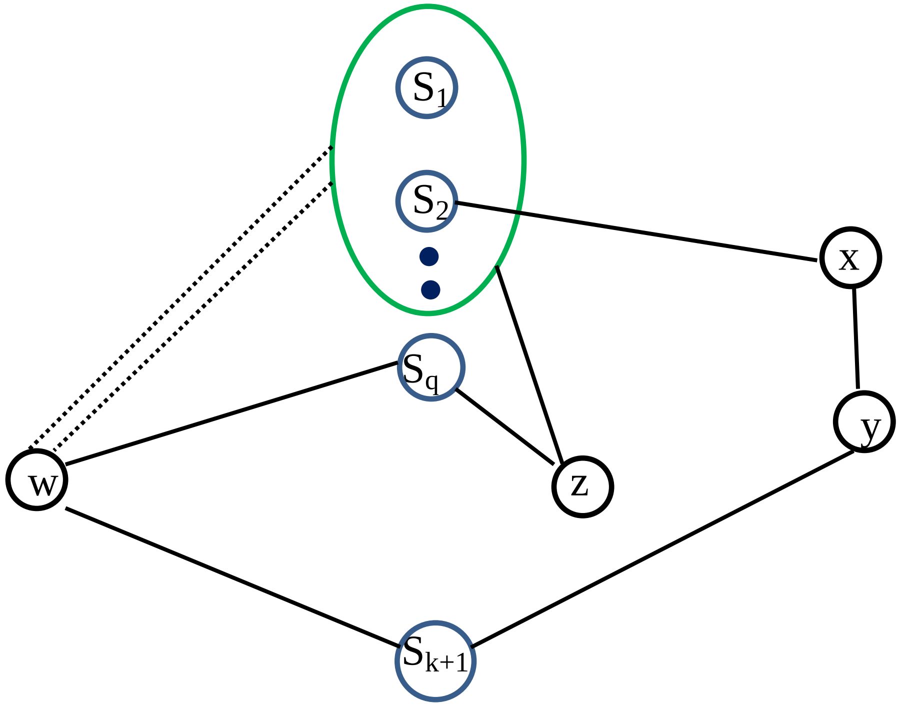
y is non-adjacent to the pseudonode and has common neighbour x with the pseudonode. By choice of s_q , z exists

Pseudonode Expansion



y is non-adjacent to the pseudonode and has common neighbour x with the pseudonode. By choice of s_q, z exists

Pseudonode Expansion



If z and x have a common neighbour in the pseudonode, we have a Meyniel obstruction......

Our algorithm:

Meyniel obstruction

G, v

strong stable set containing v

Can be apply repeatedly to give an algorithm:

Meyniel obstruction

G

clique and colouring of the same size

Previous Work

n = # vertices

m = # edges

G → Meyniel Graph Recognition Algorithm → Meyniel Obstruction

→ Promise G is Meyniel → Polytime Algorithm → Minimum Colouring

Burlet & Fonlupt (1981)

Roussel & Rusu (1999) **O(m(n+m))**

Hoàng (1987) **O(n^8)**

Hertz (1990) **O(nm)**

Roussel & Rusu (2002) **O(n^2)**

Easier

G → Polytime Algorithm → Meyniel Obstruction

→ Clique and Colouring of Same Size

KC, Lévêque, Maffray (2012) **O(n^2)**

KC, Edmonds (2005) **O(n^3)**

Algorithm (KC, Lévêque, Maffray (2012))

- Apply (slight variant of) Lexcolour Algorithm of Roussel and Rusu

- Where the colours are $C_1, \ldots, C_k$, construct a set Q as follows:
  For i=k, k-1, …, 1, let v_i be a vertex of colour i with the largest number of neighbours in Q. Add v_i to Q.
- If Q is a clique, we have a clique and colouring of the same size.
- If Q is not a clique, we can find a Meyniel obstruction.

Algorithm (KC, Lévêque, Maffray (2012))

- Apply (slight variant of) Lexcolour Algorithm of Roussel and Rusu, choosing the specified vertex to be of the first colour C_1
- Where the colours are $C_1, \ldots, C_k$, construct a set Q as follows:
  For i=k, k-1, …, 1, let v_i be a vertex of colour i with the largest number of neighbours in Q. Add v_i to Q.
- If Q is a clique, we have a clique and colouring of the same size.
- If Q is not a clique, we can find a Meyniel obstruction
- Check whether C_1 is a nice set. If not, we find a Meyniel obstruction

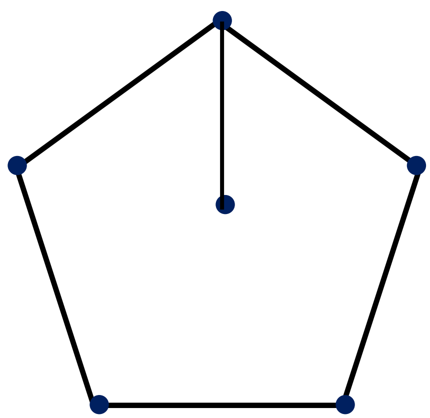
We give a polytime algorithm:

Meyniel obstruction

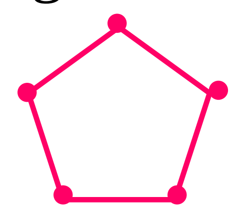
G

clique and colouring of the same size

If G contains both, we cannot predict which the algorithm will give



Meyniel obstruction



Does not have clique and colouring of the same size

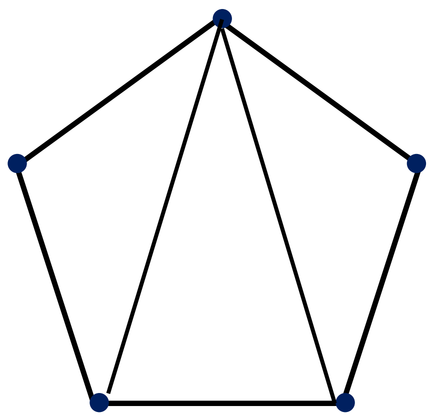
We give a polytime algorithm:

Meyniel obstruction

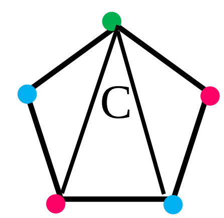
G

clique and colouring of the same size

If G contains both, we cannot predict which the algorithm will give



clique and colouring of the same size



Does not have a
Meyniel obstruction

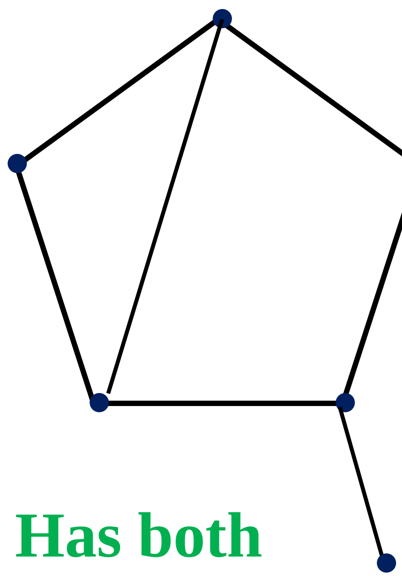
We give a polytime algorithm:

Meyniel obstruction

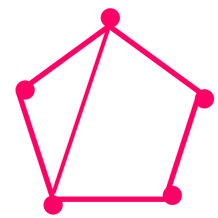
G

clique and colouring of the same size

If G contains both, we cannot predict which the algorithm will give



Meyniel obstruction



Has both

Algorithm gives one – not both

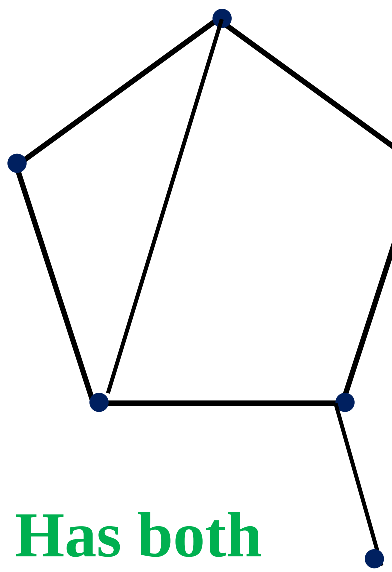
We give a polytime algorithm:

G

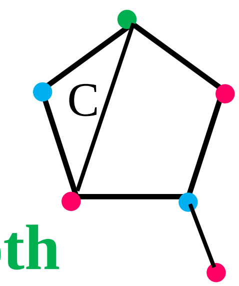
Meyniel obstruction

clique and colouring of the same size

If G contains both, we cannot predict which the algorithm will give



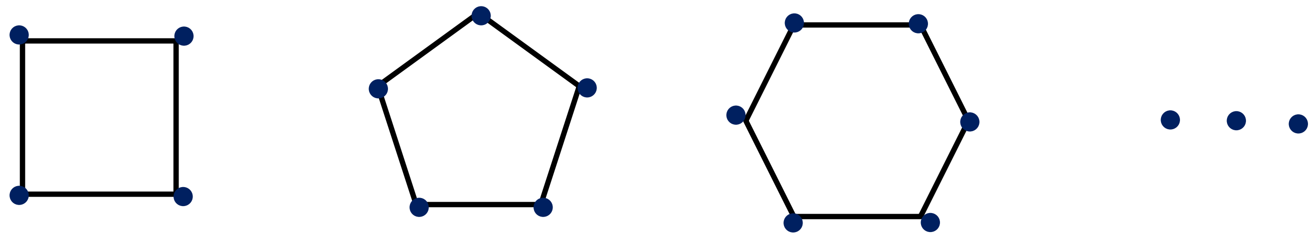
clique and colouring of the same size



Has both

Algorithm gives one – not both

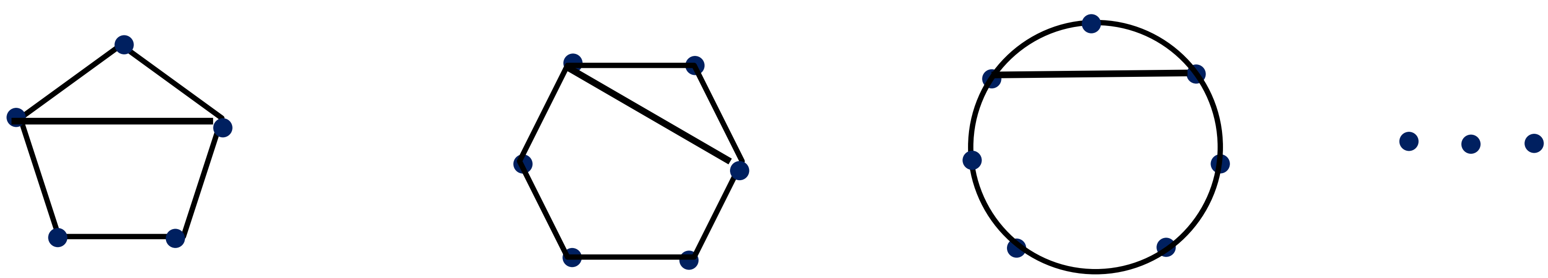
A **hole** is a chordless cycle with at least least four vertices.



C_4 C_5 C_6

A hole is **odd** or **even** depending on whether it has an odd or even number of vertices.

A **cap** consists of a hole together with an additional vertex which creates a triangle with the hole.



Meyniel graphs are the **(cap, odd hole)-free graphs**.

Meyniel graphs are the **(cap, odd hole)-free graphs**.

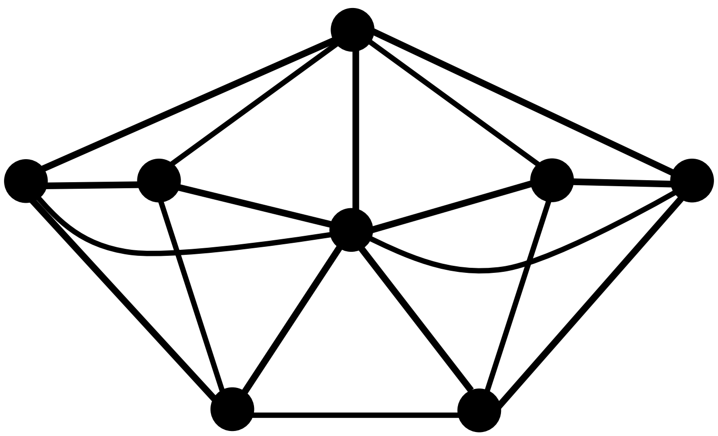
With **Kristina Vušković**, University of Leeds, Leeds, United Kingdom
Murilo da Silva, Federal University of Technology, Curitba, Brazil
Shenwei Huang, Nankai University, Tianjin, China
we have studied

(Cap, even hole)-free graphs

We obtained

- Structural results
- Chi-bound: $\chi(G) \leq (3/2)\,\omega(G)$
- O(nm) algorithms for q-colouring and max weight stable set
- polytime algorithm for minimum colouring
- Hadwiger's Conjecture holds

Theorem. KC, Huang, Da Silva, Vušković (2018)
Let **G** be a (cap, 4-hole)-free graph that contains a hole and has no clique cutset.
Let **F** be any maximal induced subgraph of G with at least 3 vertices that is triangle-free and and has no clique cutset.
Then **G** is obtained from **F** by **blowing vertices of F into cliques** and **then adding a universal clique.**

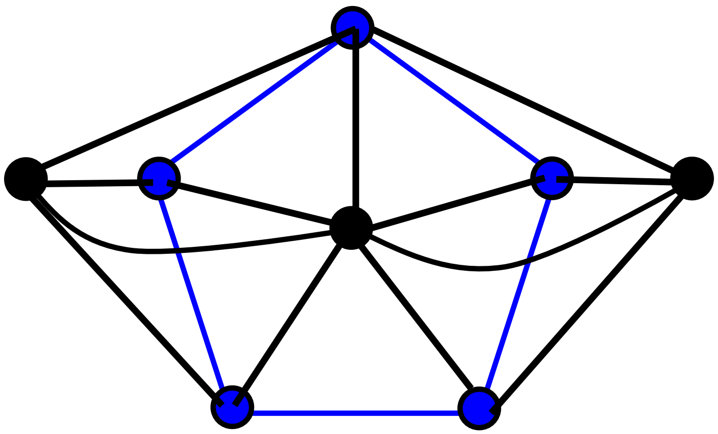


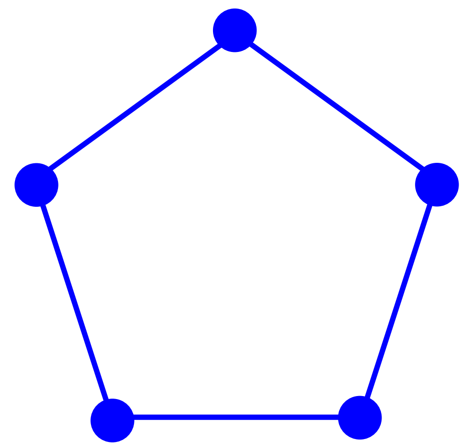
Theorem. KC, Huang, Da Silva, Vušković (2018)

Let **G** be a (cap, 4-hole)-free graph that contains a hole and has no clique cutset.

Let **F** be any maximal induced subgraph of G with at least 3 vertices that is triangle-free and and has no clique cutset.

Then **G** is obtained from **F** by **blowing vertices of F into cliques** and **then adding a universal clique.**

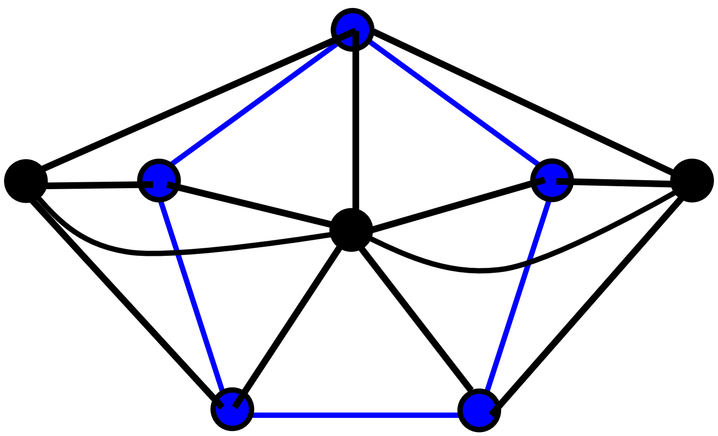


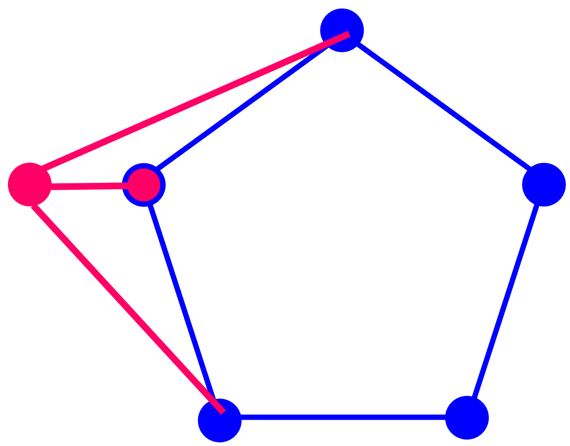


F is called the skeleton of G

Theorem. KC, Huang, Da Silva, Vušković (2018)

Let **G** be a (cap, 4-hole)-free graph that contains a hole and has no clique cutset.
Let **F** be any maximal induced subgraph of G with at least 3 vertices that is triangle-free and and has no clique cutset.
Then **G** is obtained from **F** by **blowing vertices of F into cliques** and **then adding a universal clique.**

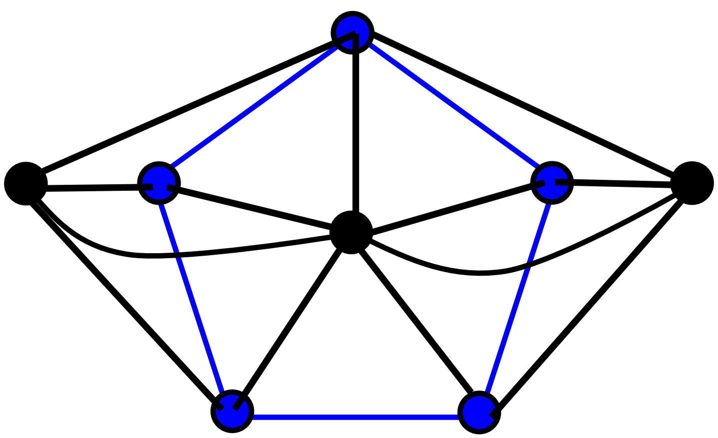


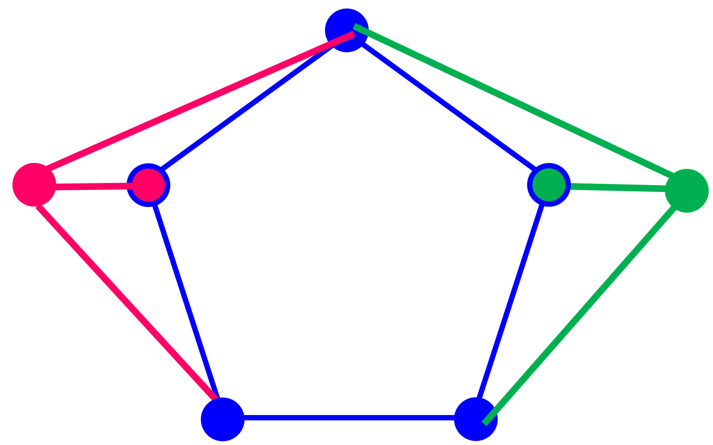


F is called the skeleton of G

Theorem. KC, Huang, Da Silva, Vušković (2018)

Let **G** be a (cap, 4-hole)-free graph that contains a hole and has no clique cutset.
Let **F** be any maximal induced subgraph of G with at least 3 vertices that is triangle-free and and has no clique cutset.
Then **G** is obtained from **F** by **blowing vertices of F into cliques** and **then adding a universal clique.**

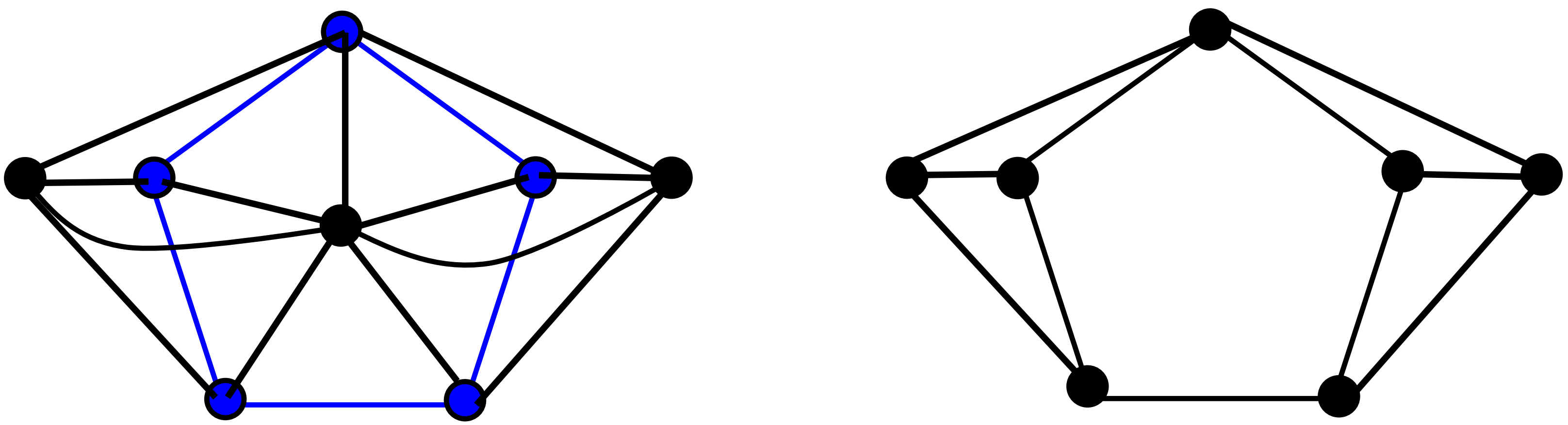




F is called the skeleton of G

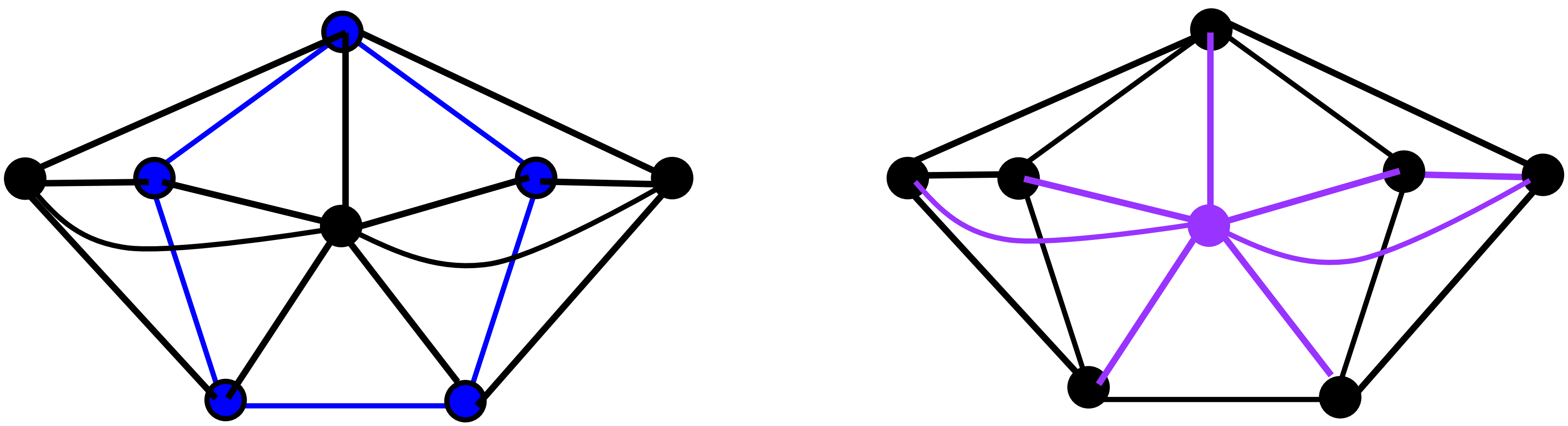
Theorem. KC, Huang, Da Silva, Vušković (2018)

Let **G** be a (cap, 4-hole)-free graph that contains a hole and has no clique cutset.
Let **F** be any maximal induced subgraph of G with at least 3 vertices that is triangle-free and and has no clique cutset.
Then **G** is obtained from **F** by **blowing vertices of F into cliques** and **then adding a universal clique.**



Theorem. KC, Huang, Da Silva, Vušković (2018)

Let **G** be a (cap, 4-hole)-free graph that contains a hole and has no clique cutset.
Let **F** be any maximal induced subgraph of G with at least 3 vertices that is triangle-free and and has no clique cutset.
Then **G** is obtained from **F** by **blowing vertices of F into cliques** and **then adding a universal clique.**



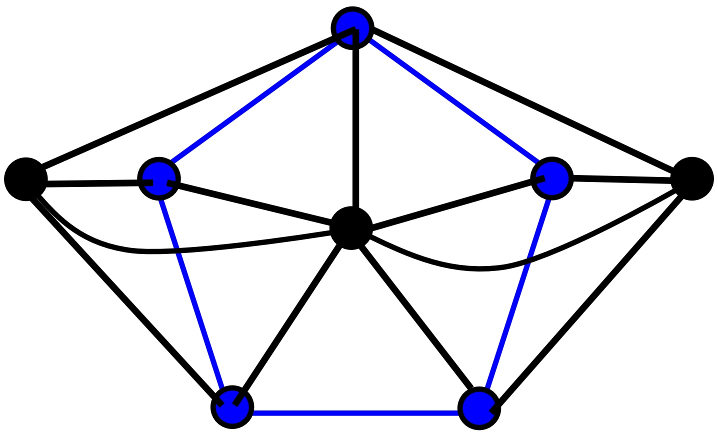
Theorem. KC, Huang, Da Silva, Vušković (2018)

Let **G** be a (cap, 4-hole)-free graph that contains a hole and has no clique cutset.

Let **F** be any maximal induced subgraph of G with at least 3 vertices that is triangle-free and and has no clique cutset.

Then **G** is obtained from **F** by **blowing vertices of F into cliques** and **then adding a universal clique.**

Further, any graph obtained by this sequence of operations starting from a (triangle, 4-hole)-free graph with at least 3 vertices and no clique cutset is (cap, 4-hole)-free and has no clique cutset.



F is called the skeleton of G

A **minor** of a graph G is obtained from a subgraph of G by contracting edges.

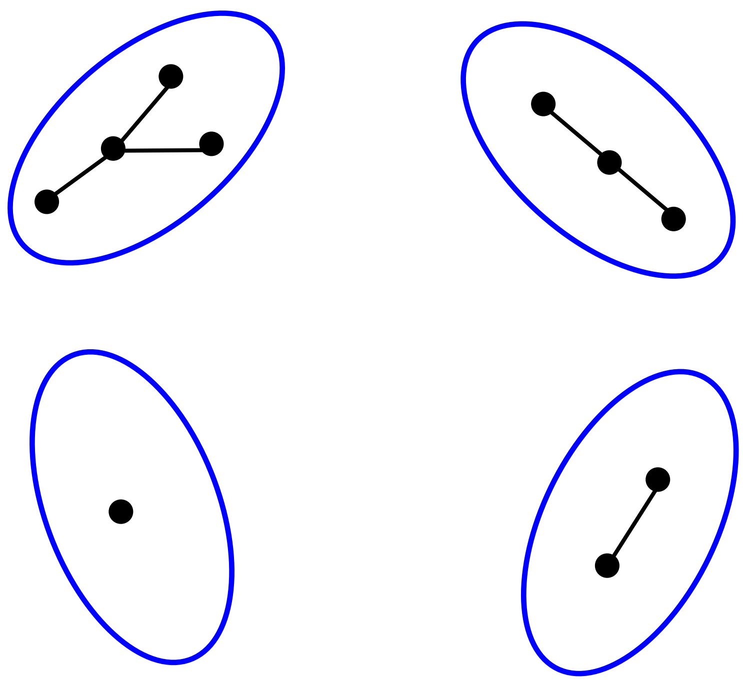
A **minor** of a graph G is obtained from a subgraph of G by contracting edges.

One way to think of a K_{t+1} minor is:

A **minor** of a graph G is obtained from a subgraph of G by contracting edges.

One way to think of a K_{t+1} minor is:

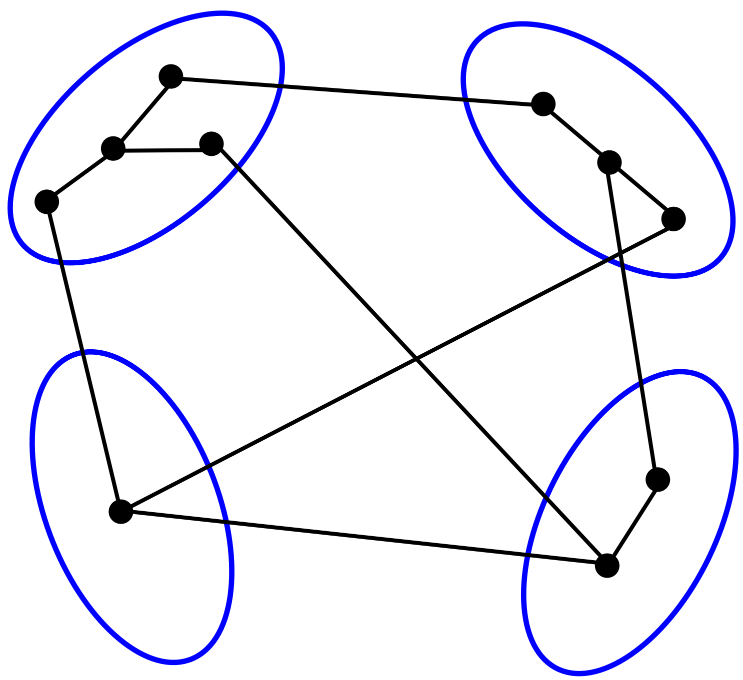
t+1 pairwise vertex-disjoint connected subgraphs



A **minor** of a graph G is obtained from a subgraph of G by contracting edges.

One way to think of a K_{t+1} minor is:

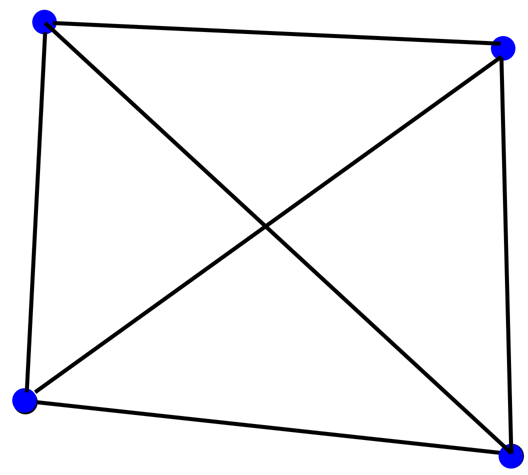
t+1 pairwise vertex-disjoint connected subgraphs
which are pairwise adjacent



A **minor** of a graph G is obtained from a subgraph of G by contracting edges.

One way to think of a K_{t+1} minor is:

t+1 pairwise vertex-disjoint connected subgraphs
which are pairwise adjacent



A **minor** of a graph G is obtained from a subgraph of G by contracting edges.

Hadwiger's Conjecture (1943)
For every integer $t \geq 0$, every graph with no K_{t+1} minor is t-colourable.

HC holds for $t \leq 5$ and remains open for $t \geq 6$:

- No K_2-minor → edgeless → 1-colourable
- No K_3-minor → no cycles → 2-colourable

- Hadwiger proved the conjecture for t = 3.
  No K_4-minor→ series-parallel→ ∃ a vertex of degree ≤ 2 → 3-colourable

- For t=4, it is equivalent to the Four Colour Theorem (Wagner 1937)

- Robertson, Seymour and Thomas (1993) proved it for t=5, using the 4CT.
  A contraction-critical 6-chromatic graph G other than K_6 has a vertex x such that G\x is planar, and is thus 4-colourable. So G is 5-colourable.

A **minor** of a graph G is obtained from a subgraph of G by contracting edges.

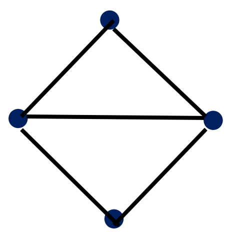
Hadwiger's Conjecture (1943)
For every integer $t \geq 0$, every graph with no K_{t+1} minor is t-colourable.

HC holds for hereditary classes χ-bounded by function $f(x)=x+1$

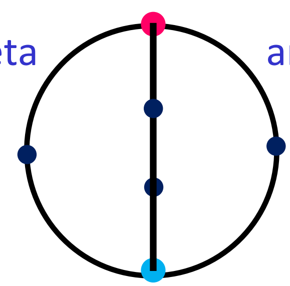
(that is, for each graph G in the class, $\chi(G) \leq \omega(G)+1$)

- perfect graphs
- line-graphs of (simple) graphs [by Vizing's Theorem]
- (theta, wheel)-free graphs [$\chi(G) \leq \max\{3, \omega(G)\}$] Radovanović, Trotignon, Vušković
- unichord-free graphs [$\chi(G) \leq \max\{3, \omega(G)\}$] Trotignon, Vušković
- (diamond, even hole)-free graphs
  [always have a vertex that is simplicial or of degree 2] Kloks, Mūller, Vušković
- (triangle, theta)-free graphs [are 3-colourable] Radovanović, Vušković
- (triangle, induced subdivision of K_4)-free graphs [are 3-colourable]
  Chudnovsky, Liu, Schaudt, Spirkl, Trotignon, Vušković

Diamond



Theta



any two of the ●-● paths form a hole

Hadwiger's Conjecture (1943)
For every integer $t \geq 0$, every graph with no K_{t+1} minor is t-colourable.

HC holds for:

- quasi-line graphs — Chudnovsky, Fradkin (2008)
  which include proper circular-arc graphs (circular interval graphs)
- graphs without a hole with size between 4 and $2\alpha(G)$ — X. Song, B. Thomas (2016)
- (C_4, C_5, P_7)-free graphs — Via structural result of KC, Huang, Penev, Sivaraman (2017+)
- (pan, even hole)-free graphs — Via structural result of KC, Chaplick, Hoàng (2018)
- (cap, even hole)-free graphs — KC, Vušković

Hadwiger's Conjecture (1943)
For every integer $t \geq 0$, every graph with no K_{t+1} minor is t-colourable.

HC holds for:

- quasi-line graphs — Chudnovsky, Fradkin (2008)
  which include proper circular-arc graphs (circular interval graphs)
- graphs without a hole with size between 4 and $2\alpha(G)$ — X. Song, B. Thomas (2016)
- (C_4, C_5, P_7)-free graphs — Via structural result of KC, Huang, Penev, Sivaraman (2017+)
- (pan, even hole)-free graphs — Via structural result of KC, Chaplick, Hoàng (2018)
- (cap, even hole)-free graphs — KC, Vušković

If a (C_4, C_5, P_7)-free graph has no induced C_7, then it is perfect.
Otherwise, it either has a clique-cutset or is a clique or has at most one non-trivial anticomponent which is a proper circular-arc graph

Hadwiger's Conjecture (1943)
For every integer $t \geq 0$, every graph with no K_{t+1} minor is t-colourable.

HC holds for:

- quasi-line graphs — Chudnovsky, Fradkin (2008)
  which include proper circular-arc graphs (circular interval graphs)
- graphs without a hole with size between 4 and $2\alpha(G)$ — X. Song, B. Thomas (2016)
- (C_4, C_5, P_7)-free graphs — Via structural result of KC, Huang, Penev, Sivaraman (2017+)
- (pan, even hole)-free graphs — Via structural result of KC, Chaplick, Hoàng (2018)
- (cap, even hole)-free graphs — KC, Vušković

A (pan, even hole)-free graph either has a clique-cutset or is a clique
or has at most one non-trivial anticomponent which is a unit circular-arc graph

Theorem: Hadwiger's Conjecture holds for (cap, even hole)-free graphs

KC + Vušković (2018+)

Proof is based on:

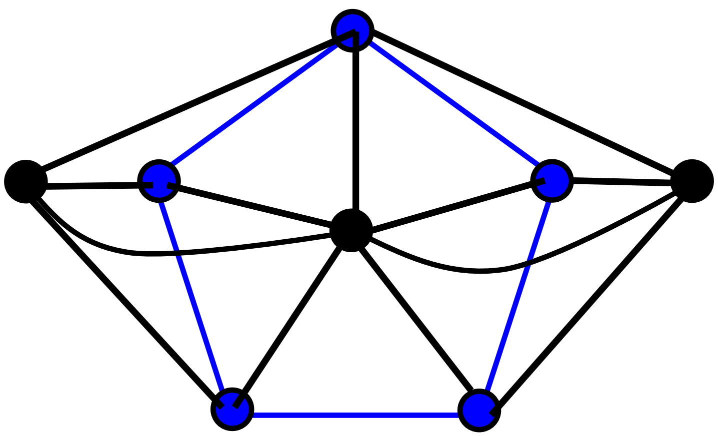
(1) Recall: Theorem KC, Huang, Da Silva,Vušković (2018)

Let **G** be a (cap, 4-hole)-free graph that contains a hole and has no clique cutset.

Let **F** be any maximal induced subgraph of G with at least 3 vertices that is triangle-free and and has no clique cutset.

Then **G** is obtained from **F** by **blowing vertices of F into cliques** and **then adding a universal clique.**

Further, any graph obtained by this sequence of operations starting from a (triangle, 4-hole)-free graph with at least 3 vertices and no clique cutset is (cap, 4-hole)-free and has no clique cutset.



(2) Lemma Conforti, Cornuéjols, Kapoor, Vušković (2000)

Every (triangle, even hole)-free graph has a vertex of degree at most 2.