Online Chromatic Number is PSPACE-Complete

Martin Böhm, Pavel Veselý

Computer Science Institute of Charles University, Prague, Czech Republic.

IWOCA 2016, August 17

Online vertex coloring

- Vertices of G are revealed one by one;

Online vertex coloring

- Vertices of G are revealed one by one;
- Algorithm sees edges to previously revealed vertices.

Online vertex coloring

- Vertices of G are revealed one by one;
- Algorithm sees edges to previously revealed vertices.
- Algorithm must color v immediately after arrival:
    - Two adjacent vertices cannot have the same color;
    - Algorithm cannot change the color later.

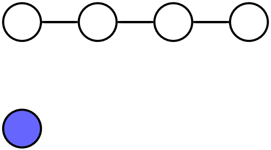
Online vertex coloring

- Vertices of G are revealed one by one;
- Algorithm sees edges to previously revealed vertices.
- Algorithm must color v immediately after arrival:
    - Two adjacent vertices cannot have the same color;
    - Algorithm cannot change the color later.
- **Goal:** Minimize no. of colors.

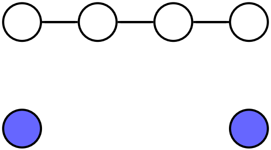
Online vertex coloring

- Vertices of G are revealed one by one;
- Algorithm sees edges to previously revealed vertices.
- Algorithm must color v immediately after arrival:
  - Two adjacent vertices cannot have the same color;
  - Algorithm cannot change the color later.
- **Goal:** Minimize no. of colors.
- **Our stronger model:** Algorithm gets a copy of G at the start.

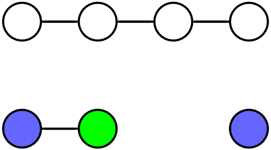
An example



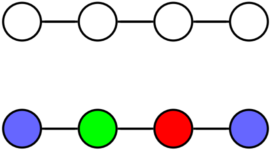
An example



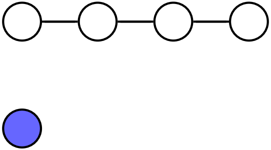
An example



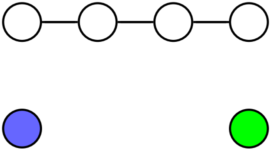
An example



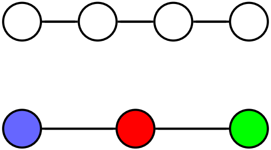
An example



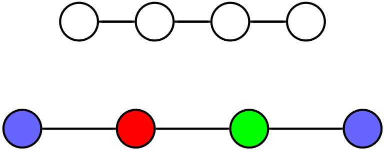
An example



An example



An example



Online chromatic number

- $\chi^O(G) =$ minimum number of colors such that a deterministic online algorithm colors G *for any input permutation of vertices*.

Online chromatic number

- $\chi^O(G)$ = minimum number of colors such that a deterministic online algorithm colors G *for any input permutation of vertices.*
- $\chi^O(P_4) = 3$.

Online chromatic number

- $\chi^O(G) =$ minimum number of colors such that a deterministic online algorithm colors G *for any input permutation of vertices*.
- $\chi^O(P_4) = 3$.
- χ^O is an (offline) graph paratemer – like chrom. number!

Online chromatic number

- $\chi^O(G) =$ minimum number of colors such that a deterministic online algorithm colors G *for any input permutation of vertices*.
- $\chi^O(P_4) = 3$.
- χ^O is an (offline) graph paratemer – like chrom. number!

History

- Online graph coloring first appears in [Bean '76].
- Online chromatic number first appears in [Gyárfás, Lehel '90].
- A copy of a graph at the start – [Halldórsson '96].

Game view

- Two players: DRAWER and PAINTER
- Both have a copy of G
- Both know a number k

Game view

- Two players: Drawer and Painter
- Both have a copy of G
- Both know a number k
  - Drawer (Sketcher) chooses the next vertex for Painter.
  - Painter paints a presented vertex with a color

Game view

- Two players: DRAWER and PAINTER
- Both have a copy of G
- Both know a number k
    - DRAWER (SKETCHER) chooses the next vertex for PAINTER.
    - PAINTER paints a presented vertex with a color
- DRAWER wins when PAINTER uses $k+1$ colors
- Otherwise PAINTER wins
- PAINTER has a winning strategy $\Leftrightarrow \chi^{O}(G) \leq k$

Complexity

Chromatic number: deciding $\chi(G) \leq k$ is NP-hard

- already for $k = 3$ colors.

Complexity

Chromatic number: deciding $\chi(G) \leq k$ is NP-hard

- already for $k = 3$ colors.

What about $\chi^O(G) \leq k$?

- In P for $k = 3$ [Gyárfás, Kiraly, Lehel '93]

Complexity

Chromatic number: deciding $\chi(G) \leq k$ is NP-hard

- already for $k = 3$ colors.

What about $\chi^O(G) \leq k$?

- In P for $k = 3$ [Gyárfás, Kiraly, Lehel '93]

[Kudahl '15]: $\chi^O(G) \leq k$ in PSPACE and coNP-hard;

- Conjecture: PSPACE-complete

Complexity and our contribution

Chromatic number: deciding $\chi(G) \leq k$ is NP-hard

- already for $k = 3$ colors.

What about $\chi^O(G) \leq k$?

- In P for $k = 3$ [Gyárfás, Kiraly, Lehel '93]

[Kudahl '15]: $\chi^O(G) \leq k$ in PSPACE and coNP-hard;

- Conjecture: PSPACE-complete

Our contribution: $\chi^O(G) \leq k$ is PSPACE-complete.

Starting step: Precoloring

Precoloring: Some vertices precolored and revealed to the algorithm at the start.

Starting step: Precoloring

Precoloring: Some vertices precolored and revealed to the algorithm at the start.

Theorem (Kudahl '15)

It is PSPACE-complete to decide $\chi^O(G) \leq k$ given that polynomially many vertices are precolored.

Three steps to the theorem

Theorem

It is PSPACE-complete to decide whether $\chi^{O}(G) \leq k$.

Three steps to the theorem

Theorem

It is PSPACE-complete to decide whether $\chi^O(G) \leq k$.

Step 1:

...with polynomially many precolored vertices (new proof).

Three steps to the theorem

Theorem
It is PSPACE-complete to decide whether $\chi^O(G) \leq k$.

Step 1:
... with polynomially many precolored vertices (new proof).

Step 2:
... with logarithmically many precolored vertices.

Three steps to the theorem

Theorem

It is PSPACE-complete to decide whether $\chi^O(G) \leq k$.

Step 1:

...with polynomially many precolored vertices (new proof).

Step 2:

...with logarithmically many precolored vertices.

Step 3:

...with no precolored vertices.

Three steps to the theorem

Theorem

It is PSPACE-complete to decide whether $\chi^O(G) \leq k$.

Step 1:

. . . with polynomially many precolored vertices (new proof).

Step 2:

. . . with logarithmically many precolored vertices.

Step 3:

. . . with no precolored vertices.

A PSPACE-Complete Problem

The problem we reduce to: Q3DNF-SAT.

- **Input:** a fully quantified 3DNF formula.
- **Question:** Is it satisfiable? $\iff$ Is at least one clause satisfiable?

$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$

Why is it PSPACE-complete?

A PSPACE-Complete Problem

The problem we reduce to: Q3DNF-SAT.

- **Input:** a fully quantified 3DNF formula.
- **Question:** Is it satisfiable? $\iff$ Is at least one clause satisfiable?

$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$

Why is it PSPACE-complete?

1. Q3CNF-SAT is PSPACE-complete (well-known).

A PSPACE-Complete Problem

The problem we reduce to: Q3DNF-SAT.

- **Input:** a fully quantified 3DNF formula.
- **Question:** Is it satisfiable? $\iff$ Is at least one clause satisfiable?

$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$

Why is it PSPACE-complete?

1. Q3CNF-SAT is PSPACE-complete (well-known).
2. PSPACE is closed under complement.

A PSPACE-Complete Problem

The problem we reduce to: Q3DNF-SAT.

- **Input:** a fully quantified 3DNF formula.
- **Question:** Is it satisfiable? $\iff$ Is at least one clause satisfiable?

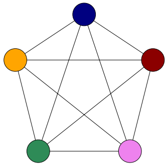
$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$

Why is it PSPACE-complete?

1. Q3CNF-SAT is PSPACE-complete (well-known).
2. PSPACE is closed under complement.
3. Q3DNF-SAT is the complement of Q3CNF-SAT.

Step 1: Precoloring

We precolor a big clique K_{col} with $|V(K_{col})| \approx$ the size of the formula.

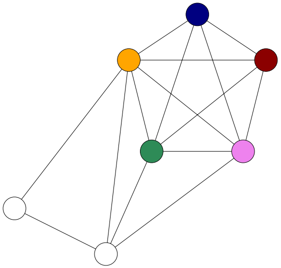


Step 1: Precoloring

We precolor a big clique K_{col} with $|V(K_{col})| \approx$ the size of the formula.

Three uses:

1. Identify uncolored vertices to player PAINTER;

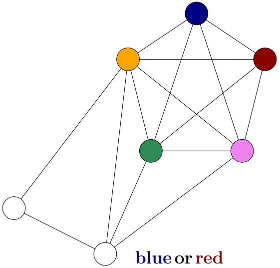


Step 1: Precoloring

We precolor a big clique K_{col} with $|V(K_{col})| \approx$ the size of the formula.

Three uses:

1. Identify uncolored vertices to player PAINTER;
2. Limit allowed colors on any uncolored vertex.

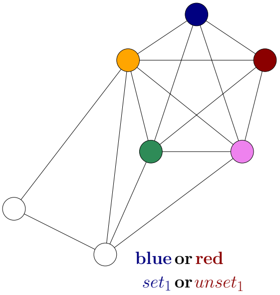


Step 1: Precoloring

We precolor a big clique K_{col} with $|V(K_{col})| \approx$ the size of the formula.

Three uses:

1. Identify uncolored vertices to player PAINTER;
2. Limit allowed colors on any uncolored vertex.
3. Assign meaning to colors.



Step 1: Reduction

Given a Q3-DNF formula

$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$

Step 1: Reduction

Given a Q3-DNF formula

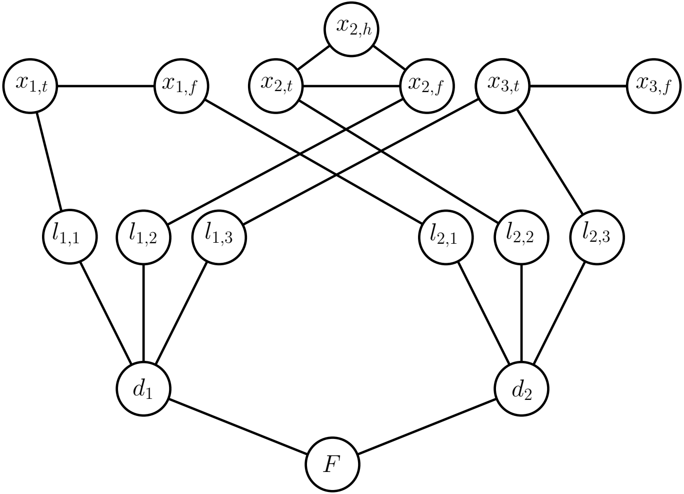
$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$

take a polynomially sized K_{col} and add gadgets for each

- variable,
- clause,
- and the entire formula.

Step 1: Gadget sketch

$$\forall x_1 \exists x_2 \forall x_3 : (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$



Step 1: Variable gadget

$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \land \neg x_2 \land x_3) \lor (\neg x_1 \land x_2 \land x_3)$$

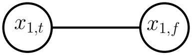
1. *Universal quantifier:* two vertices $x_{1,t}$, $x_{1,f}$. Only two allowed colors: set_1, $unset_1$.
   $x_{1,t}$ has color $set_1 \Rightarrow x_1$ set to True.

Step 1: Variable gadget

$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$

1. *Universal quantifier:* two vertices $x_{1,t}$, $x_{1,f}$. Only two allowed colors: *set*$_1$, *unset*$_1$.
   $x_{1,t}$ has color *set*$_1$ $\Rightarrow$ x_1 set to True.

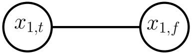
   Requirement: PAINTER cannot distinguish $x_{1,t}$ from $x_{1,f}$



Step 1: Variable gadget

$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$

1. *Universal quantifier:* two vertices $x_{1,t}$, $x_{1,f}$. Only two allowed colors: *set*$_1$, *unset*$_1$.
   $x_{1,t}$ has color *set*$_1$ $\Rightarrow$ x_1 set to True.

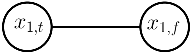
   Requirement: PAINTER cannot distinguish $x_{1,t}$ from $x_{1,f}$ ... but PAINTER knows it is coloring the gadget for x_1.

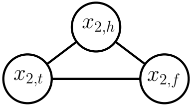
Step 1: Variable gadget

$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \land \neg x_2 \land x_3) \lor (\neg x_1 \land x_2 \land x_3)$$

1. *Universal quantifier:* two vertices $x_{1,t}$, $x_{1,f}$. Only two allowed colors: set_1, $unset_1$.
   $x_{1,t}$ has color $set_1 \Rightarrow x_1$ set to True.

   Requirement: Painter cannot distinguish $x_{1,t}$ from $x_{1,f}$ …but Painter knows it is coloring the gadget for x_1.





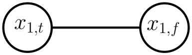
2. *Existential quantifier:* Triangle $x_{2,t}$, $x_{2,f}$ and $x_{2,h}$ – third vertex a *helper* vertex.

Step 1: Variable gadget

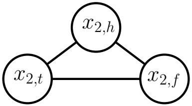
$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$

1. *Universal quantifier:* two vertices $x_{1,t}$, $x_{1,f}$. Only two allowed colors: set_1, $unset_1$.
   $x_{1,t}$ has color $set_1 \Rightarrow x_1$ set to True.

   Requirement: Painter cannot distinguish $x_{1,t}$ from $x_{1,f}$ ... but Painter knows it is coloring the gadget for x_1.



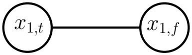
2. *Existential quantifier:* Triangle $x_{2,t}$, $x_{2,f}$ and $x_{2,h}$ – third vertex a *helper* vertex.
   Requirement: Painter **must be able** to distinguish $x_{2,t}$ from $x_{2,f}$



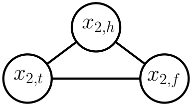
Step 1: Variable gadget

$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$

1. *Universal quantifier:* two vertices $x_{1,t}$, $x_{1,f}$. Only two allowed colors: *set*$_1$, *unset*$_1$.
   $x_{1,t}$ has color *set*$_1$ $\Rightarrow$ x_1 set to True.

   Requirement: Painter cannot distinguish $x_{1,t}$ from $x_{1,f}$ ... but Painter knows it is coloring the gadget for x_1.

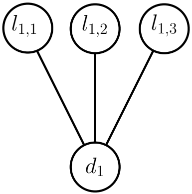
   

2. *Existential quantifier:* Triangle $x_{2,t}$, $x_{2,f}$ and $x_{2,h}$ – third vertex a *helper* vertex.
   Requirement: Painter **must be able** to distinguish $x_{2,t}$ from $x_{2,f}$ ... so all three vertices have different allowed colors.

Step 1: Clause gadgets

$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$

Literal vertex: Two allowed colors, depending on the quantifier. One of the colors always f_a.

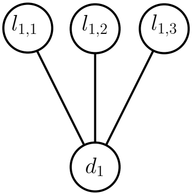


Step 1: Clause gadgets

$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$

Literal vertex: Two allowed colors, depending on the quantifier. One of the colors always f_a.

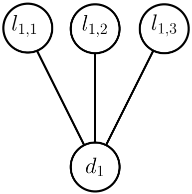
Requirement: If the literal is unsatisfied, literal vertex *must use* f_a.
Otherwise: uses the other color, f_a is available later.



Step 1: Clause gadgets

$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$

Literal vertex: Two allowed colors, depending on the quantifier. One of the colors always f_a.



Requirement: If the literal is unsatisfied, literal vertex *must use* f_a.
Otherwise: uses the other color, f_a is available later.

Clause vertex: Only two allowed colors, one of them is f_a.

Step 1: Clause gadgets

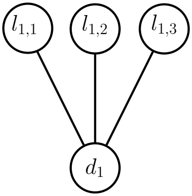
$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$

Literal vertex: Two allowed colors, depending on the quantifier. One of the colors always f_a.

Requirement: If the literal is unsatisfied, literal vertex *must use* f_a.
Otherwise: uses the other color, f_a is available later.

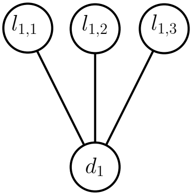
Clause vertex: Only two allowed colors, one of them is f_a.

Requirement: All three literals satisfied $\Rightarrow$ color f_a available.



Step 1: Clause gadgets

$$\forall x_1 \exists x_2 \forall x_3\colon (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$



Literal vertex: Two allowed colors, depending on the quantifier. One of the colors always f_a.

Requirement: If the literal is unsatisfied, literal vertex *must use* f_a.
Otherwise: uses the other color, f_a is available later.

Clause vertex: Only two allowed colors, one of them is f_a.

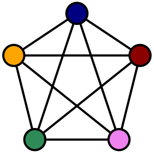
Requirement: All three literals satisfied $\Rightarrow$ color f_a available.

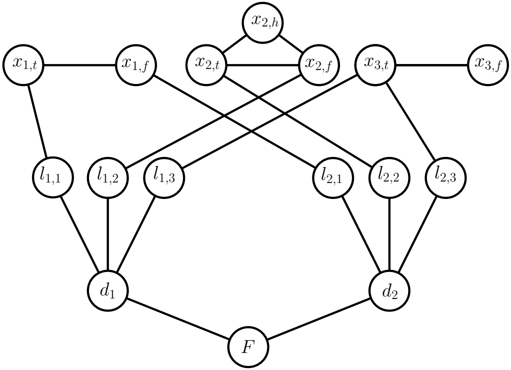
Final vertex F corresponding to the evaluation of ϕ

- We show: F can be colored $\Leftrightarrow$ ϕ is satisfiable

Step 1: Big picture

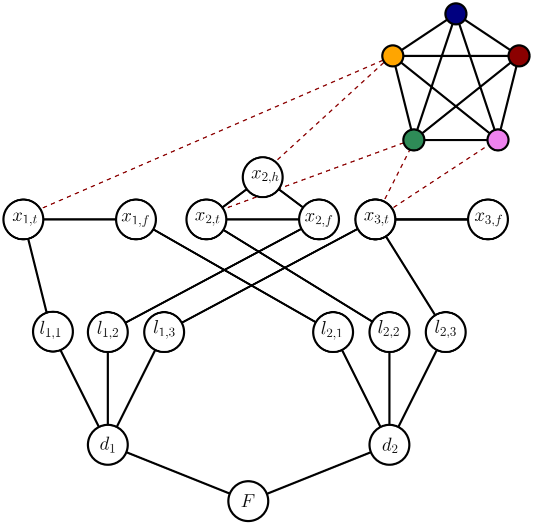
$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$





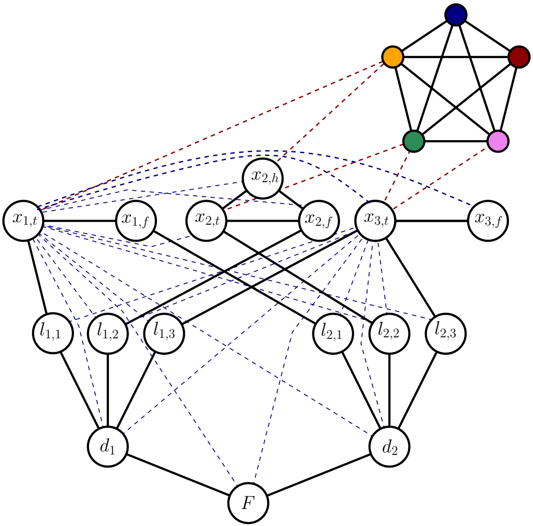
Step 1: Big picture

$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$



Step 1: Big picture

$$\forall x_1 \exists x_2 \forall x_3 \colon (x_1 \land \neg x_2 \land x_3) \lor (\neg x_1 \land x_2 \land x_3)$$



Step 1: Analysis

- If the formula is not satisfiable, then DRAWER can force PAINTER to use more than k colors

Step 1: Analysis

- If the formula is not satisfiable, then DRAWER can force PAINTER to use more than k colors
  - DRAWER presents vertices for variables in the right order

Step 1: Analysis

- If the formula is not satisfiable, then DRAWER can force PAINTER to use more than k colors
  - DRAWER presents vertices for variables in the right order
- If the formula is satisfiable, then PAINTER can color the graph using k colors

Step 1: Analysis

- If the formula is not satisfiable, then DRAWER can force PAINTER to use more than k colors
  - DRAWER presents vertices for variables in the right order
- If the formula is satisfiable, then PAINTER can color the graph using k colors
  - PAINTER can recognize vertices by allowed colors

Step 1: Analysis

- If the formula is not satisfiable, then DRAWER can force PAINTER to use more than k colors
  - DRAWER presents vertices for variables in the right order
- If the formula is satisfiable, then PAINTER can color the graph using k colors
  - PAINTER can recognize vertices by allowed colors
    - With the exception of vertices of a universally quantified variable

Step 1: Analysis

- If the formula is not satisfiable, then DRAWER can force PAINTER to use more than k colors
  - DRAWER presents vertices for variables in the right order
- If the formula is satisfiable, then PAINTER can color the graph using k colors
  - PAINTER can recognize vertices by allowed colors
    - With the exception of vertices of a universally quantified variable
  - Right order of vertices $\Rightarrow$ PAINTER uses the satisfiability
  - Bad order only helps PAINTER

Three steps to the theorem

Theorem

It is PSPACE-complete to decide whether $\chi^O(G) \leq k$.

Step 1:

…with polynomially many precolored vertices (new proof).

Step 2:

…with logarithmically many precolored vertices.

Step 3:

…with no precolored vertices.

Step 2: Sketch

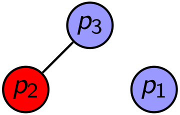
The big clique K_{col} present but uncolored.

Step 2: Sketch

The big clique K_{col} present but uncolored.

Add one *node* for every "Step 1" vertex.

Single node:



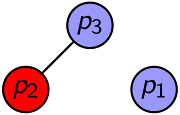
- Nodes arrive early: PAINTER uses nodes for recognition
- Nodes arrive after gadgets: PAINTER saves many colors

Step 2: Sketch

The big clique K_{col} present but uncolored.

Add one *node* for every "Step 1" vertex.

Single node:



- Nodes arrive early: PAINTER uses nodes for recognition
- Nodes arrive after gadgets: PAINTER saves many colors

Each vertex v from gadgets connected to p_1 and p_2 of each node

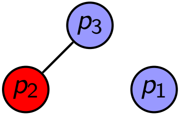
- If a node identifies v, all of its vertices are adjacent to v

Step 2: Sketch

The big clique K_{col} present but uncolored.

Add one *node* for every "Step 1" vertex.

Single node:



- Nodes arrive early: PAINTER uses nodes for recognition
- Nodes arrive after gadgets: PAINTER saves many colors

Each vertex v from gadgets connected to p_1 and p_2 of each node

- If a node identifies v, all of its vertices are adjacent to v

Only $O(\log n)$ precolored vertices

- to recognize n nodes using binary encoding.

Three steps to the theorem

Theorem

It is PSPACE-complete to decide whether $\chi^O(G) \leq k$.

Step 1:

...with polynomially many precolored vertices (new proof).

Step 2:

...with logarithmically many precolored vertices.

Step 3:

...with no precolored vertices.

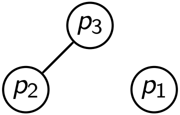
Step 3: Sketch

- Remove one precolored vertex after another;
- Graph size multiplies by a constant C;
- Total increase $C^{\log n}$ – polynomial.

Step 3: Sketch

- Remove one precolored vertex after another;
- Graph size multiplies by a constant C;
- Total increase $C^{\log n}$ – polynomial.

Idea: Supernode

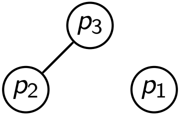


- **Difference:** p_1, p_2, p_3 are now huge cliques.
- The rest of the graph is C-times smaller.

Step 3: Sketch

- Remove one precolored vertex after another;
- Graph size multiplies by a constant C;
- Total increase $C^{\log n}$ – polynomial.

Idea: Supernode



- **Difference:** p_1, p_2, p_3 are now huge cliques.
- The rest of the graph is C-times smaller.

- Either a significant part of the supernode arrives . . .
- . . . or PAINTER can save so many colors the rest can be colored arbitrarily.

Thank you!