Unconditional Lower Bounds for Algebraic Approaches to Graph Colouring

Jakob Nordström

University of Copenhagen and Lund University

Celebration of Complexity
Royal Swedish Academy of Sciences
June 12, 2026

Based on joint work with Jonas Conneryd, Susanna de Rezende, Shuo Pang, and Kilian Risse

Facets of Complexity

Facets of Johan

Facets of Johan

Conditional results

- Postulate P $\neq$ NP or similar assumptions
- Use this to prove (extremely) strong computational infeasibility results
- Obtain web of (highly plausible) interconnected claims

Facets of Johan

Conditional results

- Postulate P $\neq$ NP or similar assumptions
- Use this to prove (extremely) strong computational infeasibility results
- Obtain web of (highly plausible) interconnected claims

Unconditional results

- Very hard for Turing machines, so look at bounded computational models

Facets of Johan

Conditional results

- Postulate P $\neq$ NP or similar assumptions
- Use this to prove (extremely) strong computational infeasibility results
- Obtain web of (highly plausible) interconnected claims

Unconditional results

- Very hard for Turing machines, so look at bounded computational models
- Circuit complexity (for restricted types of circuits)
- Communication complexity

Facets of Johan

Conditional results

- Postulate P $\neq$ NP or similar assumptions
- Use this to prove (extremely) strong computational infeasibility results
- Obtain web of (highly plausible) interconnected claims

Unconditional results

- Very hard for Turing machines, so look at bounded computational models
- Circuit complexity (for restricted types of circuits)
- Communication complexity
- Proof complexity

Facets of Johan

Conditional results

- Postulate P $\neq$ NP or similar assumptions
- Use this to prove (extremely) strong computational infeasibility results
- Obtain web of (highly plausible) interconnected claims

Unconditional results

- Very hard for Turing machines, so look at bounded computational models
- Circuit complexity (for restricted types of circuits)
- Communication complexity
- Proof complexity

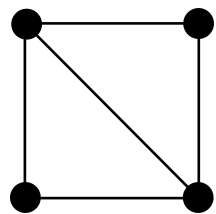
"Real-world" computation results

- Proof complexity can model algorithmic methods of reasoning
- Tools for proving lower (and upper) bounds for concrete, applied algorithms

Graph Colouring

Can vertices of graph G be coloured with k colours so that all neighbours get distinct colours?

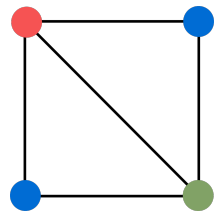
One of Karp's 21 NP-complete problems



Graph Colouring

Can vertices of graph G be coloured with k colours so that all neighbours get distinct colours?

One of Karp's 21 NP-complete problems



Is Graph Colouring Hard?

Colouring seems hard even to approximate:

- If G k-colourable, best efficient algorithm uses $k \cdot \widetilde{\Omega}(n)$ colours [Halldorsson 93]
- If G 3-colourable, best algorithm uses $n^{0.199\cdots}$ colours [Kawarabayashi–Thorup 17]
- NP-hard to approximate within factor $n^{1-\varepsilon}$ [Feige–Kilian 98, Zuckerman 07]

Is Graph Colouring Hard?

Colouring seems hard even to approximate:

- If G k-colourable, best efficient algorithm uses $k \cdot \widetilde{\Omega}(n)$ colours [Halldorsson 93]
- If G 3-colourable, best algorithm uses $n^{0.199\cdots}$ colours [Kawarabayashi–Thorup 17]
- NP-hard to approximate within factor $n^{1-\varepsilon}$ [Feige–Kilian 98, Zuckerman 07]

However, applied algorithms appear to do well:

- Backtracking and SAT-based algorithms
  [San Segundo 12, Hebrard–Katsirelos 20, Heule–Karahalios–van Hoeve 22]
- Integer programming
  [Mehortra–Trick 95, Gualandi–Malucelli 12]
- Algebraic algorithms
  [DeLoera–Lee–Malkin–Margulies 08 & 11, DeLoera–Lee–Margulies–Onn 09, DeLoera–Margulies–Pernpeinter–Riedl–Rolnick–Spencer–Stasi–Swenson 15]

Is Graph Colouring Hard?

Colouring seems hard even to approximate:

- If G k-colourable, best efficient algorithm uses $k \cdot \widetilde{\Omega}(n)$ colours [Halldorsson 93]
- If G 3-colourable, best algorithm uses $n^{0.199\cdots}$ colours [Kawarabayashi–Thorup 17]
- NP-hard to approximate within factor $n^{1-\varepsilon}$ [Feige–Kilian 98, Zuckerman 07]

However, applied algorithms appear to do well:

- Backtracking and SAT-based algorithms
  [San Segundo 12, Hebrard–Katsirelos 20, Heule–Karahalios–van Hoeve 22]
- Integer programming
  [Mehortra–Trick 95, Gualandi–Malucelli 12]
- Algebraic algorithms
  [DeLoera–Lee–Malkin–Margulies 08 & 11, DeLoera–Lee–Margulies–Onn 09, DeLoera–Margulies–Pernpeinter–Riedl–Rolnick–Spencer–Stasi–Swenson 15]

Can we prove unconditionally that graph colouring is hard for these algorithms?

Hardness for Algebraic Algorithms

- Exponential lower bounds known for explicit graphs

  [Lauria–Nordström 17, Atserias–Ochremiak 19]

- But obtained by reduction from other problems
- Graph colouring instances somewhat artificial

Hardness for Algebraic Algorithms

- Exponential lower bounds known for explicit graphs

  [Lauria–Nordström 17, Atserias–Ochremiak 19]

- But obtained by reduction from other problems
- Graph colouring instances somewhat artificial

Perhaps graph colouring is *easy on most graphs?*

Hardness for Algebraic Algorithms

- Exponential lower bounds known for explicit graphs
  [Lauria–Nordström 17, Atserias–Ochremiak 19]
- But obtained by reduction from other problems
- Graph colouring instances somewhat artificial

Perhaps graph colouring is *easy on most graphs?*

To rule this out, want average-case hardness results

SAT-based algorithms [Beame–Culberson–Mitchell–Moore 05]

Conflict-driven clause learning (CDCL) SAT solvers need exponential time for k-colouring on random graphs for $k \geq 3$

Main Focus of This Presentation

Theorem [Conneryd–deRezende–Nordström–Pang–Risse 23]

Algorithms based on Hilbert's Nullstellensatz and/or Gröbner bases require exponential time to solve k-colouring on random graphs for $k \geq 3$

Main Focus of This Presentation

Theorem [Conneryd–deRezende–Nordström–Pang–Risse 23]

Algorithms based on Hilbert's Nullstellensatz and/or Gröbner bases require exponential time to solve k-colouring on random graphs for $k \geq 3$

Established via proof complexity:

- Formalise reasoning method in algorithm as a proof system
- Fast execution for graph G with chromatic number $\chi(G) > k$
  $\Longrightarrow$ short proof of statement "G is not k-colourable"
- Show that such short proofs do not exist

Nullstellensatz Proof System

To show polynomials $p_1, \ldots, p_m$ in $\mathbb{F}[\vec{x}]$ have no common root in $\mathbb{F}$, suffices to find polynomials $q_1, \ldots, q_m$ in $\mathbb{F}[\vec{x}]$ such that

$$\sum_{i=1}^{m} q_i(\vec{x}) \cdot p_i(\vec{x}) = 1$$

This is a Nullstellensatz proof of unsatisfiability

[Beame–Impagliazzo–Krajíček–Pitassi–Pudlák 96]

Nullstellensatz Proof System

To show polynomials $p_1, \ldots, p_m$ in $\mathbb{F}[\vec{x}]$ have no common root in $\mathbb{F}$, suffices to find polynomials $q_1, \ldots, q_m$ in $\mathbb{F}[\vec{x}]$ such that

$$\sum_{i=1}^{m} q_i(\vec{x}) \cdot p_i(\vec{x}) = 1$$

This is a Nullstellensatz proof of unsatisfiability

[Beame–Impagliazzo–Krajíček–Pitassi–Pudlák 96]

Soundness: if such polynomials q_i exist, then clearly $\{p_i\}$ have no common root

Completeness (Boolean variables): special case of Hilbert's Nullstellensatz

Polynomial Calculus Proof System [Clegg–Edmonds–Impagliazzo 96]

Dynamic version: given $\{p_1, \ldots, p_m\}$, derive new polynomials using two rules

$$\text{(linear combination)} \quad \frac{p \quad q}{\alpha p + \beta q} \qquad \alpha, \beta \in \mathbb{F}$$

$$\text{(multiplication)} \quad \frac{p}{x \cdot p} \qquad x \text{ variable}$$

Goal: derive polynomial 1

Polynomial Calculus Proof System [Clegg–Edmonds–Impagliazzo 96]

Dynamic version: given $\{p_1, \ldots, p_m\}$, derive new polynomials using two rules

$$\text{(linear combination)} \quad \frac{p \quad q}{\alpha p + \beta q} \qquad \alpha, \beta \in \mathbb{F}$$

$$\text{(multiplication)} \quad \frac{p}{x \cdot p} \qquad x \text{ variable}$$

Goal: derive polynomial 1

Polynomial calculus proof system models Gröbner basis computations

Polynomial Calculus Proof System [Clegg–Edmonds–Impagliazzo 96]

Dynamic version: given $\{p_1, \ldots, p_m\}$, derive new polynomials using two rules

$$\text{(linear combination)} \quad \frac{p \quad q}{\alpha p + \beta q} \qquad \alpha, \beta \in \mathbb{F}$$

$$\text{(multiplication)} \quad \frac{p}{x \cdot p} \qquad x \text{ variable}$$

Goal: derive polynomial 1

Polynomial calculus proof system models Gröbner basis computations

- Polynomial = linear combination of monomials
- **Proof size:** # monomials in derivation
- **Proof degree:** max total degree of polynomial in derivation

Encoding k-Colouring as Polynomials

Variables $x_{v,i}$ = "vertex v gets colour i", $v \in V(G)$, $i \in [k]$

Axiom polynomials for graph G:

Each vertex gets a colour	$\sum_{i=1}^{k} x_{v,i} - 1$	
Colours are unique	$x_{v,i} \cdot x_{v,i'}$	$i < i'$
Distinct colours for neighbours	$x_{u,i} \cdot x_{v,i}$	$(u,v) \in E(G)$
Variables are Boolean	$x_{v,i}^2 - x_{v,i}$	

Encoding k-Colouring as Polynomials

Variables $x_{v,i}$ = "vertex v gets colour i", $v \in V(G)$, $i \in [k]$

Axiom polynomials for graph G:

Each vertex gets a colour	$\sum_{i=1}^{k} x_{v,i} - 1$	
Colours are unique	$x_{v,i} \cdot x_{v,i'}$	$i < i'$
Distinct colours for neighbours	$x_{u,i} \cdot x_{v,i}$	$(u,v) \in E(G)$
Variables are Boolean	$x_{v,i}^2 - x_{v,i}$	

Common root of polynomials $\Leftrightarrow$ k-colouring of G

Encoding k-Colouring as Polynomials

Variables $x_{v,i}$ = "vertex v gets colour i", $v \in V(G)$, $i \in [k]$

Axiom polynomials for graph G:

Each vertex gets a colour	$\sum_{i=1}^{k} x_{v,i} - 1$	
Colours are unique	$x_{v,i} \cdot x_{v,i'}$	$i < i'$
Distinct colours for neighbours	$x_{u,i} \cdot x_{v,i}$	$(u,v) \in E(G)$
Variables are Boolean	$x_{v,i}^2 - x_{v,i}$	

Common root of polynomials $\Leftrightarrow$ k-colouring of G

Other important encoding used in computational algebra [Bayer 82]:

- Colours X_v are kth roots of unity $\{1, \zeta, \zeta^2, \cdots, \zeta^{k-1}\}$ (assuming $\mathrm{char}(\mathbb{F}) \nmid k$)
- Affine substitution from X_v to $x_{v,1}, \ldots, x_{v,k} \Rightarrow$ (roughly) same proof degree

More Formal Statement of Result

Theorem

For G random sparse graph on n vertices, with probability $1 - o(1)$ any polynomial calculus proof of fact "G is not 3-colourable" has size $\exp(\Omega(n))$

More Formal Statement of Result

Theorem

For G random sparse graph on n vertices, with probability $1 - o(1)$ any polynomial calculus proof of fact "G is not 3-colourable" has size $\exp(\Omega(n))$

- Lower bound holds over any field
- For both random regular graphs and Erdős–Rényi random graphs (with appropriately chosen parameters)

More Formal Statement of Result

Theorem

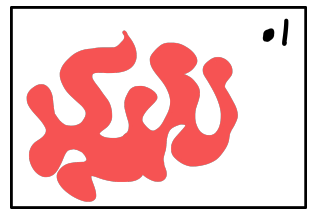
For G random sparse graph on n vertices, with probability $1 - o(1)$ any polynomial calculus proof of fact "G is not 3-colourable" has size $\exp(\Omega(n))$

- Lower bound holds over any field
- For both random regular graphs and Erdős–Rényi random graphs (with appropriately chosen parameters)
- Obtained by showing $\Omega(n)$ degree lower bound
- Implies exponential size lower bound for Boolean encoding

[Impagliazzo–Pudlák–Sgall 99]

Degree Lower Bound: Framework

Task: **separate** 1 from $\{$polynomials derivable in degree $D\}$

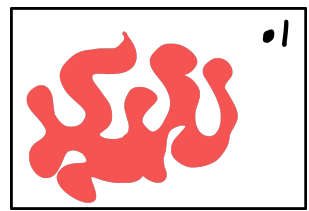


Degree Lower Bound: Framework

Task: **separate** 1 from $\{$polynomials derivable in degree $D\}$

[Razborov 98]: suffices to find linear $R : \mathbb{F}[\vec{x}] \to \mathbb{F}[\vec{x}]$ such that

1. $R(\text{axiom}) = 0$
2. $R(xp) = R(xR(p))$ for any p of degree $\leq D - 1$
3. $R(1) \neq 0$



Derivable in degree D

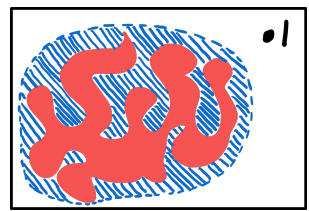
Degree Lower Bound: Framework

Task: **separate** 1 from $\{$polynomials derivable in degree $D\}$

[Razborov 98]: suffices to find linear $R : \mathbb{F}[\vec{x}] \to \mathbb{F}[\vec{x}]$ such that

1. $R(\text{axiom}) = 0$
2. $R(xp) = R(xR(p))$ for any p of degree $\leq D - 1$
3. $R(1) \neq 0$

Kernel of R overapproximates what is derivable in degree D



Derivable in degree D

$\ker(R)$

Quick Recap: Polynomial Ideals

Ideal $\langle \mathcal{P} \rangle$ generated by set of polynomials $\mathcal{P}$ is smallest set such that

- $\mathcal{P} \subseteq \langle \mathcal{P} \rangle$
- $p, q \in \langle \mathcal{P} \rangle \Rightarrow p + q \in \langle \mathcal{P} \rangle$
- $p \in \langle \mathcal{P} \rangle \Rightarrow r \cdot p \in \langle \mathcal{P} \rangle$ for all polynomials r

Quick Recap: Polynomial Ideals

Ideal $\langle \mathcal{P} \rangle$ generated by set of polynomials $\mathcal{P}$ is smallest set such that

- $\mathcal{P} \subseteq \langle \mathcal{P} \rangle$
- $p, q \in \langle \mathcal{P} \rangle \Rightarrow p + q \in \langle \mathcal{P} \rangle$
- $p \in \langle \mathcal{P} \rangle \Rightarrow r \cdot p \in \langle \mathcal{P} \rangle$ for all polynomials r

Connection to polynomial calculus:

- $\langle \mathcal{P} \rangle$ contains all polynomial implied by $\mathcal{P}$
- Which is exactly what is derivable by polynomial calculus
- $1 \in \langle \mathcal{P} \rangle \Leftrightarrow \langle \mathcal{P} \rangle$ = all polynomials $\Leftrightarrow \mathcal{P}$ is unsatisfiable

Polynomial Ideal Reductions

- Impose total order on monomials (with 1 smallest)
- Order polynomials by largest monomial (leading monomial)
- **Reduction modulo ideal $\langle \mathcal{P} \rangle$:** Operator $R_{\langle \mathcal{P} \rangle} : \mathbb{F}[\vec{x}] \to \mathbb{F}[\vec{x}]$ defined as

$$R_{\langle \mathcal{P} \rangle}(q) := \text{minimum polynomial in } \{q - p \mid p \in \langle \mathcal{P} \rangle\}$$

Polynomial Ideal Reductions

- Impose total order on monomials (with 1 smallest)
- Order polynomials by largest monomial (leading monomial)
- **Reduction modulo ideal $\langle \mathcal{P} \rangle$:** Operator $R_{\langle \mathcal{P} \rangle} : \mathbb{F}[\vec{x}] \to \mathbb{F}[\vec{x}]$ defined as

$$R_{\langle \mathcal{P} \rangle}(q) := \text{minimum polynomial in } \{q - p \mid p \in \langle \mathcal{P} \rangle\}$$

Properties of $R_{\langle \mathcal{P} \rangle}$:

- well-defined
- linear
- $\ker\left(R_{\langle \mathcal{P} \rangle}\right) = \langle \mathcal{P} \rangle$

Pseudo-Reductions

Reduction operator $R_{\langle \mathcal{P} \rangle}$ satisfies properties postulated by Razborov!

Pseudo-Reductions

Reduction operator $R_{\langle \mathcal{P} \rangle}$ satisfies properties postulated by Razborov!
Except $R_{\langle \mathcal{P} \rangle}(1) = 0$, since $\mathcal{P}$ unsatisfiable...
So won't get degree lower bounds from reduction modulo $\langle \mathcal{P} \rangle$

Pseudo-Reductions

Reduction operator $R_{\langle \mathcal{P} \rangle}$ satisfies properties postulated by Razborov!
Except $R_{\langle \mathcal{P} \rangle}(1) = 0$, since $\mathcal{P}$ unsatisfiable...
So won't get degree lower bounds from reduction modulo $\langle \mathcal{P} \rangle$

Fix: reduce modulo smaller ideals!

Alekhnovich-Razborov 03

- Reduce each monomial m modulo ideal of subset $S(m)$ of axioms
- Extend to polynomials by linearity

Pseudo-Reductions

Reduction operator $R_{\langle \mathcal{P} \rangle}$ satisfies properties postulated by Razborov!
Except $R_{\langle \mathcal{P} \rangle}(1) = 0$, since $\mathcal{P}$ unsatisfiable...
So won't get degree lower bounds from reduction modulo $\langle \mathcal{P} \rangle$

Fix: reduce modulo smaller ideals!

Alekhnovich-Razborov 03

- Reduce each monomial m modulo ideal of subset $S(m)$ of axioms
- Extend to polynomials by linearity

Intuition:

- $S(m)$ contains axioms "closely related" to variables in m
- R indistinguishable from polynomial ideal reduction in low degree, but $R(1) \neq 0$
- Think of R as pseudo-reduction modulo fake ideal claiming that $\mathcal{P}$ is satisfiable

From Pseudo-Reductions to Degree Lower Bounds

Recall that we want three properties from linear operator R:

1. $R(\text{axiom}) = 0$
2. $R(xp) = R(xR(p))$ for any p of degree $\leq D - 1$
3. $R(1) \neq 0$

From Pseudo-Reductions to Degree Lower Bounds

Recall that we want three properties from linear operator R:

1. $R(\text{axiom}) = 0$
2. $R(xp) = R(xR(p))$ for any p of degree $\leq D - 1$
3. $R(1) \neq 0$

This would show:

- All input axioms in $\mathcal{P}$ are in $\ker(R)$
- All polynomials derivable from $\mathcal{P}$ in degree $\leq D$ are in $\ker(R)$
- But $1 \notin \ker(R)$
- So degree lower bound $> D$ follows

Getting Pseudo-Reductions to Behave Well

- Concretely, for axiom polynomial $p = m_1 + m_2$ want $R(p) = 0$

Getting Pseudo-Reductions to Behave Well

- Concretely, for axiom polynomial $p = m_1 + m_2$ want $R(p) = 0$
- But pseudo-reduction

$$R(p) = R(m_1) + R(m_2) = R_{\langle S(m_1)\rangle}(m_1) + R_{\langle S(m_2)\rangle}(m_2)$$

reduces monomials modulo different ideals — lose control of what happens

Getting Pseudo-Reductions to Behave Well

- Concretely, for axiom polynomial $p = m_1 + m_2$ want $R(p) = 0$
- But pseudo-reduction

$$R(p) = R(m_1) + R(m_2) = R_{\langle S(m_1)\rangle}(m_1) + R_{\langle S(m_2)\rangle}(m_2)$$

  reduces monomials modulo different ideals — lose control of what happens
- Dream scenario: Show that there exists ideal $\mathcal{I}$ such that
  - $p \in \mathcal{I}$ for our axiom $p = m_1 + m_2$
  - $S(m_i) \subseteq \mathcal{I}$ for $i = 1, 2$
  - $R_{\langle S(m_i)\rangle}(m_i) = R_{\mathcal{I}}(m_i)$ for $i = 1, 2$

Getting Pseudo-Reductions to Behave Well

- Concretely, for axiom polynomial $p = m_1 + m_2$ want $R(p) = 0$
- But pseudo-reduction

$$R(p) = R(m_1) + R(m_2) = R_{\langle S(m_1)\rangle}(m_1) + R_{\langle S(m_2)\rangle}(m_2)$$

  reduces monomials modulo different ideals — lose control of what happens
- Dream scenario: Show that there exists ideal $\mathcal{I}$ such that
  - $p \in \mathcal{I}$ for our axiom $p = m_1 + m_2$
  - $S(m_i) \subseteq \mathcal{I}$ for $i = 1, 2$
  - $R_{\langle S(m_i)\rangle}(m_i) = R_{\mathcal{I}}(m_i)$ for $i = 1, 2$
- Then

$$R(p) = R_{\langle S(m_1)\rangle}(m_1) + R_{\langle S(m_2)\rangle}(m_2) = R_{\mathcal{I}}(m_1) + R_{\mathcal{I}}(m_2) = R_{\mathcal{I}}(m_1 + m_2) = 0$$

Why Aren't We Done Already?

- All of this is old news...
  - Proposed in [Alekhnovich–Razborov 03]
  - Further developed in, e.g., [Galesi–Lauria 10a, 10b; Mikša–Nordström 15]

Why Aren't We Done Already?

- All of this is old news...
  - Proposed in [Alekhnovich–Razborov 03]
  - Further developed in, e.g., [Galesi–Lauria 10a, 10b; Mikša–Nordström 15]
- Technical crux: Requires finding subset of axioms that can be "nicely isolated"
  - For, e.g., pigeonhole principle (PHP), if some pigeons assigned to holes, residual problem is still PHP instance
  - But partial colouring propagates constraints throughout whole graph!?

Why Aren't We Done Already?

- All of this is old news...
  - Proposed in [Alekhnovich–Razborov 03]
  - Further developed in, e.g., [Galesi–Lauria 10a, 10b; Mikša–Nordström 15]
- Technical crux: Requires finding subset of axioms that can be "nicely isolated"
  - For, e.g., pigeonhole principle (PHP), if some pigeons assigned to holes, residual problem is still PHP instance
  - But partial colouring propagates constraints throughout whole graph!?
- Average-case polynomial calculus lower bounds for colouring open since [Beame–Culberson–Mitchell–Moore 05]

Degree Lower Bounds for Colouring

- For colouring, associate to each monomial m a vertex set V_m
- Let $S(m)$ = axioms "induced graph $G[V_m]$ is k-colourable"
- Slightly abuse notation R_{V_m} to mean reduction modulo ideal generated by $S(m)$
- Define $R\big(\sum_i c_i m_i\big) \coloneqq \sum_i c_i R_{V_{m_i}}(m_i)$
- Technical challenge: construct V_m so that R satisfies required properties

Vertex Set V_m

Say that monomial $m = x_{u,2}x_{v,3}x_{w,1}$ mentions vertices u, v, w

Vertices V_m related to m

- Define closure $\mathrm{Cl}(U) \supseteq U$ of vertex sets U
- Set $V_m := \mathrm{Cl}(\{\text{vertices mentioned in } m\})$

Vertex Set V_m

Say that monomial $m = x_{u,2}x_{v,3}x_{w,1}$ mentions vertices u, v, w

Vertices V_m related to m

- Define closure $\mathsf{Cl}(U) \supseteq U$ of vertex sets U
- Set $V_m := \mathsf{Cl}(\{\text{vertices mentioned in } m\})$

Desired properties of closure:

1. **Subset-preserving:** $U' \subseteq \mathsf{Cl}(U) \Rightarrow \mathsf{Cl}(U') \subseteq \mathsf{Cl}(U)$
2. **Size-preserving:** $|U| \leq D \Rightarrow |\mathsf{Cl}(U)| = O(D)$
3. **Reduction-preserving:** For any monomial m mentioning only vertices in $\mathsf{Cl}(U)$ and any vertex set J of size $O(D)$ it holds that

$$R_{\mathsf{Cl}(U)}(m) = R_{\mathsf{Cl}(U)\cup J}(m)$$

Vertex Set V_m

Say that monomial $m = x_{u,2}x_{v,3}x_{w,1}$ mentions vertices u, v, w

Vertices V_m related to m

- Define closure $\mathsf{Cl}(U) \supseteq U$ of vertex sets U
- Set $V_m := \mathsf{Cl}(\{\text{vertices mentioned in } m\})$

Desired properties of closure:

1. **Subset-preserving:** $U' \subseteq \mathsf{Cl}(U) \Rightarrow \mathsf{Cl}(U') \subseteq \mathsf{Cl}(U)$
2. **Size-preserving:** $|U| \leq D \Rightarrow |\mathsf{Cl}(U)| = O(D)$
3. **Reduction-preserving:** For any monomial m mentioning only vertices in $\mathsf{Cl}(U)$ and any vertex set J of size $O(D)$ it holds that

$$R_{\mathsf{Cl}(U)}(m) = R_{\mathsf{Cl}(U)\cup J}(m)$$

Reduction-Preserving Property of Closure

Reduction-preserving: For any monomial m mentioning only vertices in $\mathsf{Cl}(U)$ and any vertex set J of size $O(D)$ it holds that $R_{\mathsf{Cl}(U)}(m) = R_{\mathsf{Cl}(U)\cup J}(m)$

Reduction-Preserving Property of Closure

Reduction-preserving: For any monomial m mentioning only vertices in $\mathsf{Cl}(U)$ and any vertex set J of size $O(D)$ it holds that $R_{\mathsf{Cl}(U)}(m) = R_{\mathsf{Cl}(U)\cup J}(m)$

Reduction lemma [CdRNPR 23]

For fixed order on vertices (and variables), can achieve this property if:

- each colouring of $G[\mathsf{Cl}(U)]$ can be extended to $G[\mathsf{Cl}(U) \cup J]$

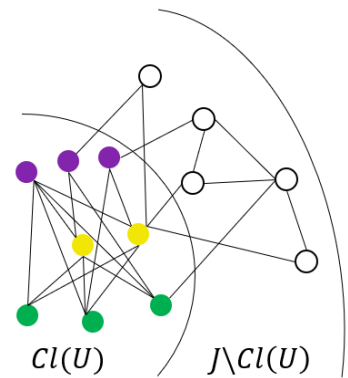
Reduction-Preserving Property of Closure

Reduction-preserving: For any monomial m mentioning only vertices in $\mathsf{Cl}(U)$ and any vertex set J of size $O(D)$ it holds that $R_{\mathsf{Cl}(U)}(m) = R_{\mathsf{Cl}(U)\cup J}(m)$

Reduction lemma [CdRNPR 23]

For fixed order on vertices (and variables), can achieve this property if:

- each colouring of $G[\mathsf{Cl}(U)]$ can be extended to $G[\mathsf{Cl}(U) \cup J]$



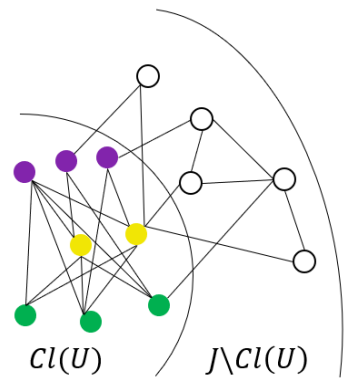
Reduction-Preserving Property of Closure

Reduction-preserving: For any monomial m mentioning only vertices in $\mathsf{Cl}(U)$ and any vertex set J of size $O(D)$ it holds that $R_{\mathsf{Cl}(U)}(m) = R_{\mathsf{Cl}(U)\cup J}(m)$

Reduction lemma [CdRNPR 23]

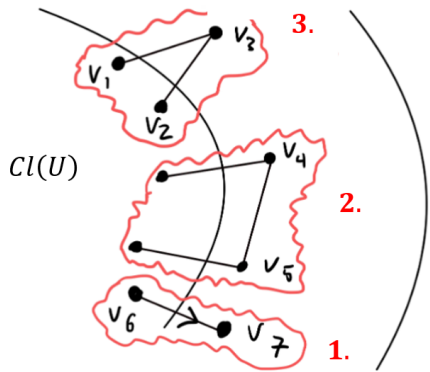
For fixed order on vertices (and variables), can achieve this property if:

- each colouring of $G[\mathsf{Cl}(U)]$ can be extended to $G[\mathsf{Cl}(U) \cup J]$
- ... in **order-decreasing** way: for each v in $J \setminus \mathsf{Cl}(U)$, colour can be determined based on colouring of $\{w \in \mathsf{Cl}(U) : w < v\}$



Construction of Closure (1/2)

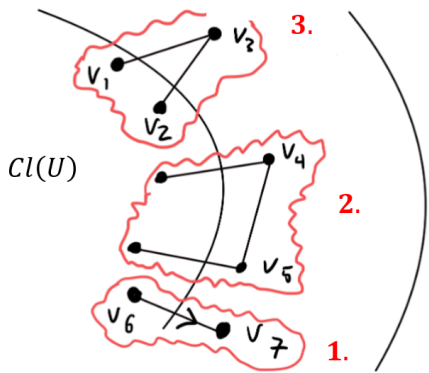
Sufficient to prevent certain structures in neighbourhood outside Cl(U) such as



Construction of Closure (1/2)

Sufficient to prevent certain structures in neighbourhood outside Cl(U) such as

1. Vertex with a larger neighbour in Cl(U)
2. Edge between neighbours of Cl(U)
3. Vertex with > 1 neighbour in Cl(U)

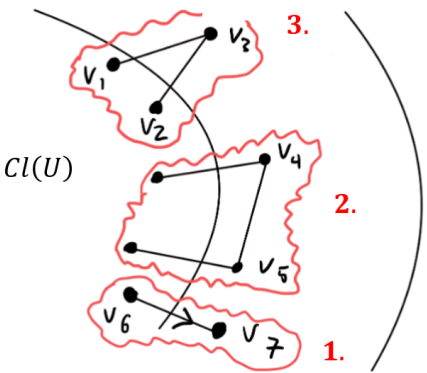


Construction of Closure (1/2)

Sufficient to prevent certain structures in neighbourhood outside Cl(U) such as

1. Vertex with a larger neighbour in Cl(U)
2. Edge between neighbours of Cl(U)
3. Vertex with > 1 neighbour in Cl(U)

Similar structures identified in [Romero-Tunçel 22] *in colouring lower bound for large-girth graphs!*

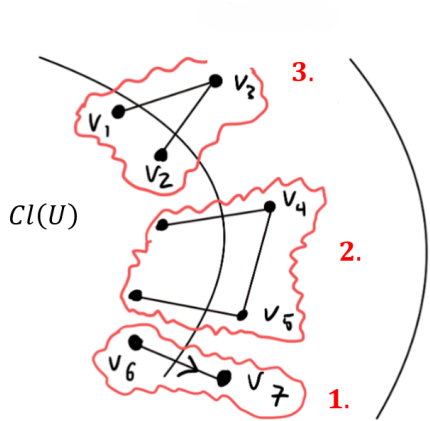


Construction of Closure (2/2)

How to prevent bad structures in neighbourhood outside Cl(U):

Constructing the closure of set U

1. Start with given set U

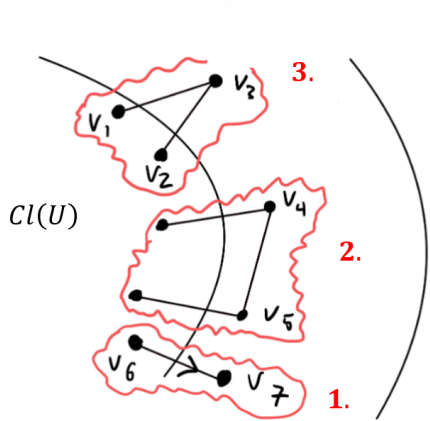


Construction of Closure (2/2)

How to prevent bad structures in neighbourhood outside Cl(U):

Constructing the closure of set U

1. Start with given set U
2. Add all vertices reachable from current set by order-decreasing paths in G

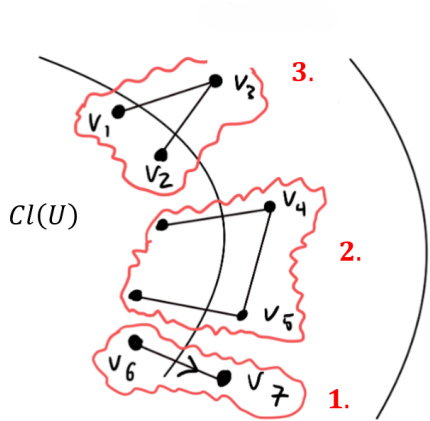


Construction of Closure (2/2)

How to prevent bad structures in neighbourhood outside Cl(U):

Constructing the closure of set U

1. Start with given set U
2. Add all vertices reachable from current set by order-decreasing paths in G
3. If type 2 or 3 structure, add offending vertices to current set and go to **2**



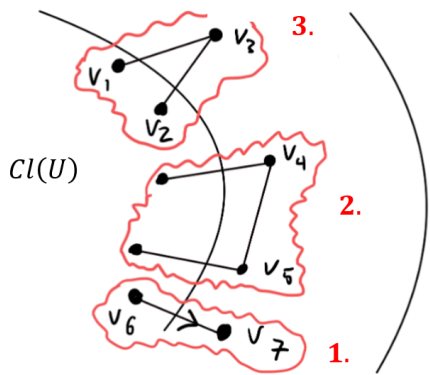
Construction of Closure (2/2)

How to prevent bad structures in neighbourhood outside Cl(U):

Constructing the closure of set U

1. Start with given set U
2. Add all vertices reachable from current set by order-decreasing paths in G
3. If type 2 or 3 structure, add offending vertices to current set and go to **2**

Let Cl(U) := final set



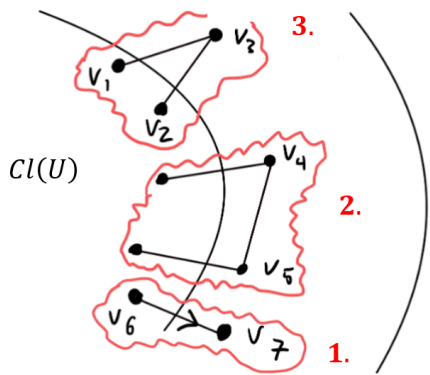
Construction of Closure (2/2)

How to prevent bad structures in neighbourhood outside Cl(U):

Constructing the closure of set U

1. Start with given set U
2. Add all vertices reachable from current set by order-decreasing paths in G
3. If type 2 or 3 structure, add offending vertices to current set and go to **2**

Let Cl(U) := final set



Not hard to show Cl(U) well-defined

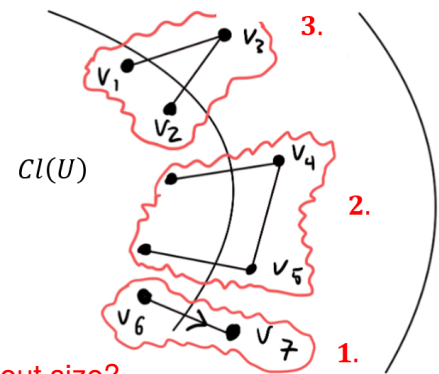
Construction of Closure (2/2)

How to prevent bad structures in neighbourhood outside Cl(U):

Constructing the closure of set U

1. Start with given set U
2. Add all vertices reachable from current set by order-decreasing paths in G
3. If type 2 or 3 structure, add offending vertices to current set and go to **2**

Let Cl(U) := final set



Not hard to show Cl(U) well-defined, but what about size?

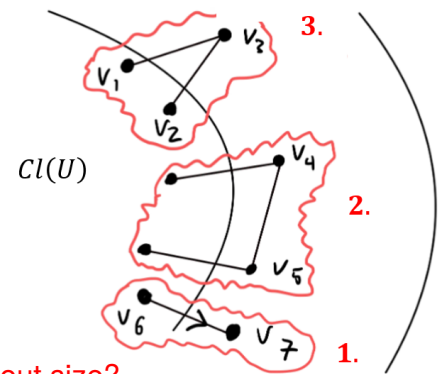
Construction of Closure (2/2)

How to prevent bad structures in neighbourhood outside Cl(U):

Constructing the closure of set U

1. Start with given set U
2. Add all vertices reachable from current set by order-decreasing paths in G
3. If type 2 or 3 structure, add offending vertices to current set and go to **2**

Let Cl(U) := final set



Not hard to show Cl(U) well-defined, but what about size?
OK for random graph if *vertices ordered according to colouring* [Romero-Tunçel 22]

Some Future Research Directions

1. Colouring lower bounds for other proof systems
   - Sherali–Adams (linear programming hierarchies)
   - Sum-of-squares (semidefinite programming hierarchies)
   - Cutting planes (integer linear programming)

Some Future Research Directions

1. Colouring lower bounds for other proof systems
   - Sherali–Adams (linear programming hierarchies)
   - Sum-of-squares (semidefinite programming hierarchies)
   - Cutting planes (integer linear programming)
2. Polynomial calculus lower bounds for other problems
   - Graph homomorphism (generalization of colouring)
   - Clique

Some Future Research Directions

1. Colouring lower bounds for other proof systems
   - Sherali–Adams (linear programming hierarchies)
   - Sum-of-squares (semidefinite programming hierarchies)
   - Cutting planes (integer linear programming)
2. Polynomial calculus lower bounds for other problems
   - Graph homomorphism (generalization of colouring)
   - Clique
3. Connections between pseudo-reductions and other lower bound operators
   - Designs for Nullstellensatz
   - Pseudo-expectations for sums-of-squares

Summing up

- Graph colouring notoriously hard problem in theory
- But applied algorithms can work surprisingly well in practice
- Can we rule out unconditionally that such algorithms solve NP-complete problems in polynomial time?

Summing up

- Graph colouring notoriously hard problem in theory
- But applied algorithms can work surprisingly well in practice
- Can we rule out unconditionally that such algorithms solve NP-complete problems in polynomial time?
- **This talk:** Linear degree lower bounds for polynomial calculus proofs for random graphs
- Implies exponential average-case running times for state-of-the-art algebraic algorithms

Summing up

- Graph colouring notoriously hard problem in theory
- But applied algorithms can work surprisingly well in practice
- Can we rule out unconditionally that such algorithms solve NP-complete problems in polynomial time?
- **This talk:** Linear degree lower bounds for polynomial calculus proofs for random graphs
- Implies exponential average-case running times for state-of-the-art algebraic algorithms
- Complexity theory as a tool for understanding “real-world” computation

Summing up

- Graph colouring notoriously hard problem in theory
- But applied algorithms can work surprisingly well in practice
- Can we rule out unconditionally that such algorithms solve NP-complete problems in polynomial time?
- **This talk:** Linear degree lower bounds for polynomial calculus proofs for random graphs
- Implies exponential average-case running times for state-of-the-art algebraic algorithms
- Complexity theory as a tool for understanding “real-world” computation

Thank you for your attention!

Summing up

- Graph colouring notoriously hard problem in theory
- But applied algorithms can work surprisingly well in practice
- Can we rule out unconditionally that such algorithms solve NP-complete problems in polynomial time?
- **This talk:** Linear degree lower bounds for polynomial calculus proofs for random graphs
- Implies exponential average-case running times for state-of-the-art algebraic algorithms
- Complexity theory as a tool for understanding “real-world” computation

Thank you for your attention! And thank you, Johan!