Proof Complexity Lower Bounds from Graph Expansion and Combinatorial Games

Jakob Nordström

KTH Royal Institute of Technology
Stockholm, Sweden

Algebraic Methods
Simons Institute for the Theory of Computing
December 3, 2018

Based on joint work with Massimo Lauria and Mladen Mikša

Proof Complexity and Expansion

- **General goal:** Prove that concrete proof systems cannot efficiently certify unsatisfiability of concrete CNF formulas

- **General theme:**

CNF formula $\mathcal{F}$ "expanding"

$\Downarrow$

Large proofs needed to refute $\mathcal{F}$

- Paradigm implemented for
  - resolution: well-developed machinery
  - polynomial calculus: very much less so

  (Will define these proof systems shortly)

- What "expanding" means is usually a formula-specific hack

Lower Bounds by Playing Games on Graphs

Given CNF formula $\mathcal{F}$ over variables $\mathcal{V}$, build bipartite graph

- Left vertex set partition of clauses into $\mathcal{F} = \bigcup_{i=1}^{m} F_i$
- Right vertex set division of variables $\mathcal{V} = \bigcup_{j=1}^{n} V_j$
- Edge (F_i, V_j) if $Vars(F_i) \cap V_j \neq \emptyset$

Lower Bounds by Playing Games on Graphs

Given CNF formula $\mathcal{F}$ over variables $\mathcal{V}$, build bipartite graph

- Left vertex set partition of clauses into $\mathcal{F} = \bigcup_{i=1}^{m} F_i$
- Right vertex set division of variables $\mathcal{V} = \bigcup_{j=1}^{n} V_j$
- Edge (F_i, V_j) if $Vars(F_i) \cap V_j \neq \emptyset$

Lower bound on proof size if

1. Bipartite graph is an expander (very well-connected)
2. We can win the edge game on every edge (F_i, V_j)

Lower Bounds by Playing Games on Graphs

Given CNF formula $\mathcal{F}$ over variables $\mathcal{V}$, build bipartite graph

- Left vertex set partition of clauses into $\mathcal{F} = \bigcup_{i=1}^{m} F_i$
- Right vertex set division of variables $\mathcal{V} = \bigcup_{j=1}^{n} V_j$
- Edge (F_i, V_j) if $\mathit{Vars}(F_i) \cap V_j \neq \emptyset$

Lower bound on proof size if

1. Bipartite graph is an expander (very well-connected)
2. We can win the edge game on every edge (F_i, V_j)

Edge game on (F_i, V_j)

- Adversary assigns all variables $\mathcal{V} \setminus V_j$
- We assign V_j
- We win if F_i true

Main Message

Edge game on (F_i, V_j)

- Adversary assigns all variables $\mathcal{V} \setminus V_j$
- We assign V_j
- We win if F_i true

Main Message

Edge game on (F_i, V_j)

- Adversary assigns all variables $\mathcal{V} \setminus V_j$
- We assign V_j
- We win if F_i true

Who goes first?

- Adversary has to start $\Rightarrow$ resolution lower bound
- We have to start $\Rightarrow$ polynomial calculus lower bound

Main Message

Edge game on (F_i, V_j)

- Adversary assigns all variables $\mathcal{V} \setminus V_j$
- We assign V_j
- We win if F_i true

Who goes first?

- Adversary has to start $\Rightarrow$ resolution lower bound
- We have to start $\Rightarrow$ polynomial calculus lower bound

Consequences

- Extends techniques in [BW01] and [AR03]
- Unifies many previous lower bounds
- And yields some new ones

Outline

Just To Make Sure We're on the Same Page...

- Literal a: variable x or its negation $\overline{x}$
- Clause $C = a_1 \vee \cdots \vee a_k$: disjunction of literals
  (Consider as sets, so no repetitions and order irrelevant)
- CNF formula $\mathcal{F} = C_1 \wedge \cdots \wedge C_m$: conjunction of clauses
- k-CNF formula: CNF formula with clauses of size $\leq k$
  $k = \mathcal{O}(1)$ constant in this talk
- $\mathit{true} = 1$; $\mathit{false} = 0$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

$$\frac{C \lor x \qquad D \lor \overline{x}}{C \lor D}$$

Refutation ends when empty clause $\bot$ derived

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

$$\frac{C \lor x \qquad D \lor \overline{x}}{C \lor D}$$

Refutation ends when empty clause $\perp$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1. $x \lor y$
2. $x \lor \overline{y} \lor z$
3. $\overline{x} \lor z$
4. $\overline{y} \lor \overline{z}$
5. $\overline{x} \lor \overline{z}$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

$$\frac{C \lor x \qquad D \lor \overline{x}}{C \lor D}$$

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \lor y$	Axiom
2.	$x \lor \overline{y} \lor z$	Axiom
3.	$\overline{x} \lor z$	Axiom
4.	$\overline{y} \lor \overline{z}$	Axiom
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$x \lor \overline{y}$	Res$(2,4)$
7.	x	Res$(1,6)$
8.	$\overline{x}$	Res$(3,5)$
9.	$\bot$	Res$(7,8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D}$$

Refutation ends when empty clause $\perp$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \vee y$	Axiom
2.	$\boldsymbol{x \vee \overline{y} \vee z}$	Axiom
3.	$\overline{x} \vee z$	Axiom
4.	$\boldsymbol{\overline{y} \vee \overline{z}}$	Axiom
5.	$\overline{x} \vee \overline{z}$	Axiom
6.	$x \vee \overline{y}$	Res$(2,4)$
7.	x	Res$(1,6)$
8.	$\overline{x}$	Res$(3,5)$
9.	$\perp$	Res$(7,8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D}$$

Refutation ends when empty clause $\perp$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \vee y$	Axiom
2.	$\boldsymbol{x \vee \overline{y} \vee z}$	Axiom
3.	$\overline{x} \vee z$	Axiom
4.	$\boldsymbol{\overline{y} \vee \overline{z}}$	Axiom
5.	$\overline{x} \vee \overline{z}$	Axiom
6.	$\boldsymbol{x \vee \overline{y}}$	Res$(2, 4)$
7.	x	Res$(1, 6)$
8.	$\overline{x}$	Res$(3, 5)$
9.	$\perp$	Res$(7, 8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

$$\frac{C \lor x \qquad D \lor \overline{x}}{C \lor D}$$

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \lor y$	Axiom
2.	$x \lor \overline{y} \lor z$	Axiom
3.	$\overline{x} \lor z$	Axiom
4.	$\overline{y} \lor \overline{z}$	Axiom
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$\boldsymbol{x \lor \overline{y}}$	Res$(2, 4)$
7.	x	Res$(1, 6)$
8.	$\overline{x}$	Res$(3, 5)$
9.	$\bot$	Res$(7, 8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

$$\frac{C \lor x \qquad D \lor \overline{x}}{C \lor D}$$

Refutation ends when empty clause $\perp$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$\boldsymbol{x \lor y}$	Axiom
2.	$x \lor \overline{y} \lor z$	Axiom
3.	$\overline{x} \lor z$	Axiom
4.	$\overline{y} \lor \overline{z}$	Axiom
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$\boldsymbol{x \lor \overline{y}}$	Res$(2,4)$
7.	x	Res$(1,6)$
8.	$\overline{x}$	Res$(3,5)$
9.	$\perp$	Res$(7,8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

$$\frac{C \lor x \qquad D \lor \overline{x}}{C \lor D}$$

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$\boldsymbol{x \lor y}$	Axiom
2.	$x \lor \overline{y} \lor z$	Axiom
3.	$\overline{x} \lor z$	Axiom
4.	$\overline{y} \lor \overline{z}$	Axiom
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$\boldsymbol{x \lor \overline{y}}$	Res$(2,4)$
7.	$\boldsymbol{x}$	Res$(1,6)$
8.	$\overline{x}$	Res$(3,5)$
9.	$\bot$	Res$(7,8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

$$\frac{C \lor x \qquad D \lor \overline{x}}{C \lor D}$$

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \lor y$	Axiom
2.	$x \lor \overline{y} \lor z$	Axiom
3.	$\overline{x} \lor z$	Axiom
4.	$\overline{y} \lor \overline{z}$	Axiom
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$x \lor \overline{y}$	Res$(2,4)$
7.	$\boldsymbol{x}$	Res$(1,6)$
8.	$\overline{x}$	Res$(3,5)$
9.	$\bot$	Res$(7,8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D}$$

Refutation ends when empty clause $\perp$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \vee y$	Axiom
2.	$x \vee \overline{y} \vee z$	Axiom
3.	$\boldsymbol{\overline{x} \vee z}$	Axiom
4.	$\overline{y} \vee \overline{z}$	Axiom
5.	$\boldsymbol{\overline{x} \vee \overline{z}}$	Axiom
6.	$x \vee \overline{y}$	Res$(2, 4)$
7.	x	Res$(1, 6)$
8.	$\overline{x}$	Res$(3, 5)$
9.	$\perp$	Res$(7, 8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D}$$

Refutation ends when empty clause $\perp$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \vee y$	Axiom
2.	$x \vee \overline{y} \vee z$	Axiom
3.	$\overline{x} \vee z$	Axiom
4.	$\overline{y} \vee \overline{z}$	Axiom
5.	$\overline{x} \vee \overline{z}$	Axiom
6.	$x \vee \overline{y}$	Res$(2, 4)$
7.	x	Res$(1, 6)$
8.	$\overline{x}$	Res$(3, 5)$
9.	$\perp$	Res$(7, 8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

$$\frac{C \lor x \qquad D \lor \overline{x}}{C \lor D}$$

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \lor y$	Axiom
2.	$x \lor \overline{y} \lor z$	Axiom
3.	$\overline{x} \lor z$	Axiom
4.	$\overline{y} \lor \overline{z}$	Axiom
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$x \lor \overline{y}$	Res$(2, 4)$
7.	x	Res$(1, 6)$
8.	$\overline{x}$	Res$(3, 5)$
9.	$\bot$	Res$(7, 8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D}$$

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \vee y$	Axiom
2.	$x \vee \overline{y} \vee z$	Axiom
3.	$\overline{x} \vee z$	Axiom
4.	$\overline{y} \vee \overline{z}$	Axiom
5.	$\overline{x} \vee \overline{z}$	Axiom
6.	$x \vee \overline{y}$	Res$(2, 4)$
7.	$\boldsymbol{x}$	Res$(1, 6)$
8.	$\boldsymbol{\overline{x}}$	Res$(3, 5)$
9.	$\bot$	Res$(7, 8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

$$\frac{C \lor x \qquad D \lor \overline{x}}{C \lor D}$$

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \lor y$	Axiom
2.	$x \lor \overline{y} \lor z$	Axiom
3.	$\overline{x} \lor z$	Axiom
4.	$\overline{y} \lor \overline{z}$	Axiom
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$x \lor \overline{y}$	Res$(2,4)$
7.	$\boldsymbol{x}$	Res$(1,6)$
8.	$\boldsymbol{\overline{x}}$	Res$(3,5)$
9.	$\bot$	Res$(7,8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D}$$

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \vee y$	Axiom
2.	$x \vee \overline{y} \vee z$	Axiom
3.	$\overline{x} \vee z$	Axiom
4.	$\overline{y} \vee \overline{z}$	Axiom
5.	$\overline{x} \vee \overline{z}$	Axiom
6.	$x \vee \overline{y}$	Res$(2,4)$
7.	x	Res$(1,6)$
8.	$\overline{x}$	Res$(3,5)$
9.	$\bot$	Res$(7,8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

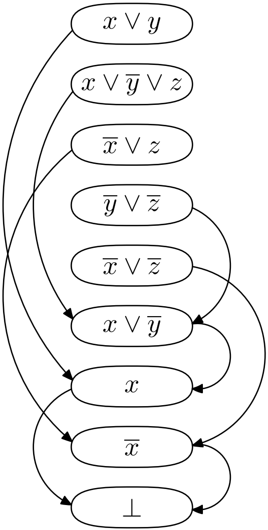
Derive new clauses by resolution rule

$$\frac{C \lor x \qquad D \lor \overline{x}}{C \lor D}$$

Refutation ends when empty clause $\perp$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph



The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

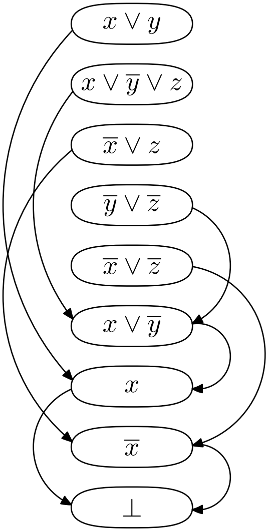
$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D}$$

Refutation ends when empty clause $\perp$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

Tree-like resolution if DAG is tree



Resolution Size/Length and Width

Size/length $= \#$ clauses in refutation [9 in our example]

Most fundamental measure in proof complexity
Never worse than $\exp(\mathcal{O}(\#\text{variables}))$
Matching $\exp(\Omega(\text{formula size}))$ lower bounds known

Resolution Size/Length and Width

Size/length = # clauses in refutation [9 in our example]

Most fundamental measure in proof complexity
Never worse than $\exp(\mathcal{O}(\#\text{variables}))$
Matching $\exp(\Omega(\text{formula size}))$ lower bounds known

Width = size of largest clause in refutation [3 in our example]

Always $\leq$ #variables
Helpful measure to get a handle on size (as we shall soon see)

Polynomial Calculus (PC)

From [CEI96]; with adjustment in [ABRW02]

Clauses interpreted as polynomials over field $\mathbb{F}$
(Evaluate to true $\equiv$ vanish)

Example: $x \vee y \vee \overline{z}$ gets translated to $\overline{x}\overline{y}z$

Polynomial Calculus (PC)

From [CEI96]; with adjustment in [ABRW02]

Clauses interpreted as polynomials over field $\mathbb{F}$
(Evaluate to true $\equiv$ vanish)

Example: $x \lor y \lor \overline{z}$ gets translated to $\overline{x}\,\overline{y}z$

Derivation rules

Boolean axioms $\dfrac{}{x^2 - x}$

Negation $\dfrac{}{x + \overline{x} - 1}$

Linear combination $\dfrac{p \quad q}{\alpha p + \beta q}$

Multiplication $\dfrac{p}{xp}$

Goal: Derive $1 \Leftrightarrow$ no common root $\Leftrightarrow$ formula unsatisfiable

Formalizes Gröbner basis computations

Polynomial Calculus Size and Degree

Clauses turn into monomials
Write out all polynomials as sums of monomials
W.l.o.g. all polynomials multilinear (because of Boolean axioms)

Polynomial Calculus Size and Degree

Clauses turn into monomials
Write out all polynomials as sums of monomials
W.l.o.g. all polynomials multilinear (because of Boolean axioms)

Size — analogue of resolution length/size
total # monomials in refutation counted with repetitions

Degree — analogue of resolution width
largest degree of monomial in refutation

Polynomial Calculus Stronger than Resolution

Polynomial calculus can simulate resolution proofs efficiently with respect to both size and width/degree

- Can mimic resolution refutation step by step
- Hence worst-case upper bounds for resolution carry over

Polynomial Calculus Stronger than Resolution

Polynomial calculus can simulate resolution proofs efficiently with respect to both size and width/degree

- Can mimic resolution refutation step by step
- Hence worst-case upper bounds for resolution carry over

Example: Resolution step:

$$\frac{x \vee \overline{y} \vee z \qquad \overline{y} \vee \overline{z}}{x \vee \overline{y}}$$

Polynomial Calculus Stronger than Resolution

Polynomial calculus can simulate resolution proofs efficiently with respect to both size and width/degree

- Can mimic resolution refutation step by step
- Hence worst-case upper bounds for resolution carry over

Example: Resolution step:

$$\frac{x \lor \overline{y} \lor z \qquad \overline{y} \lor \overline{z}}{x \lor \overline{y}}$$

simulated by polynomial calculus derivation:

$$\cfrac{\overline{x}y\overline{z} \qquad \cfrac{\cfrac{yz}{\overline{x}yz} \qquad \cfrac{\cfrac{z + \overline{z} - 1}{yz + y\overline{z} - y}}{\overline{x}yz + \overline{x}y\overline{z} - \overline{x}y}}{-\overline{x}y\overline{z} + \overline{x}y}}{\overline{x}y}$$

Examples of Some Hard Formulas (1/3)

Random k-CNF formulas

Δn randomly sampled k-clauses over n variables

($\Delta \gtrsim 4.5$ sufficient to get unsatisfiable 3-CNF almost surely)

Exponential size lower bounds for for

- resolution [CS88, BKPS02]
- polynomial calculus over fields of characteristic $\neq 2$ [BI99]
- polynomial calculus over any field [AR03]

Examples of Some Hard Formulas (2/3)

Pigeonhole principle (PHP)

"$n + 1$ pigeons don't fit into n holes"

Variables $p_{i,j}$ = "pigeon i goes into hole j"

$p_{i,1} \lor p_{i,2} \lor \cdots \lor p_{i,n}$	every pigeon i gets a hole
$\overline{p}_{i,j} \lor \overline{p}_{i',j}$	no hole j gets two pigeons $i \neq i'$

Can also add "functionality" and "onto" axioms

$\overline{p}_{i,j} \lor \overline{p}_{i,j'}$	no pigeon i gets two holes $j \neq j'$
$p_{1,j} \lor p_{2,j} \lor \cdots \lor p_{n+1,j}$	every hole j gets a pigeon

Examples of Some Hard Formulas (2/3)

Pigeonhole principle (PHP)

"$n+1$ pigeons don't fit into n holes"

Variables $p_{i,j}$ = "pigeon i goes into hole j"

$p_{i,1} \lor p_{i,2} \lor \cdots \lor p_{i,n}$	every pigeon i gets a hole
$\overline{p}_{i,j} \lor \overline{p}_{i',j}$	no hole j gets two pigeons $i \neq i'$

Can also add "functionality" and "onto" axioms

$\overline{p}_{i,j} \lor \overline{p}_{i,j'}$	no pigeon i gets two holes $j \neq j'$
$p_{1,j} \lor p_{2,j} \lor \cdots \lor p_{n+1,j}$	every hole j gets a pigeon

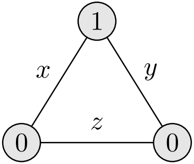
- All PHP versions exponentially hard for resolution [Hak85]
- "Vanilla PHP" exponentially hard for PC [AR03]
- Onto functional PHP easy for PC (over any field) [Rii93]
- What about functional PHP and onto PHP for PC?

Examples of Some Hard Formulas (3/3)

Tseitin formulas

"Sum of degrees of vertices in graph is even"

- Label every vertex 0/1 so that sum of labels odd
- Write CNF requiring parity of # true incident edges = label



$$
\begin{aligned}
&(x \lor y) && \land (\overline{x} \lor z) \\
\land\ &(\overline{x} \lor \overline{y}) && \land (y \lor \overline{z}) \\
\land\ &(x \lor \overline{z}) && \land (\overline{y} \lor z)
\end{aligned}
$$

- Exponentially hard for resolution on expanders [Urq87]
- And for polynomial calculus in characteristic $\neq 2$ [BGIP01]
- But PC over $\mathrm{GF}(2)$ can do Gaussian elimination

Upper Bounds from Resolution Width and PC Degree

Width/degree upper bound ⇒ size upper bound

Resolution: At most $(2 \cdot \#\text{variables})^{\text{width}}$ distinct clauses

Polynomial calculus: Essentially same bound; more careful argument [CEI96]

These simple upper bounds are essentially tight [ALN16]

Upper Bounds from Resolution Width and PC Degree

Width/degree upper bound ⇒ size upper bound

Resolution: At most $(2 \cdot \#\text{variables})^{\text{width}}$ distinct clauses

Polynomial calculus: Essentially same bound; more careful argument [CEI96]

These simple upper bounds are essentially tight [ALN16]

Width/degree lower bound ⇒ size lower bound

Much less obvious. . .

Width/Degree Lower Bounds Imply Size Lower Bounds

Theorem ([IPS99, BW01])

For k-CNF formula over N variables

$$\textit{proof size} \geq \exp\left(\Omega\left(\frac{(\textit{proof width/degree})^2}{N}\right)\right)$$

Width/Degree Lower Bounds Imply Size Lower Bounds

Theorem ([IPS99, BW01])

For k-CNF formula over N variables

$$\textit{proof size} \geq \exp\left(\Omega\left(\frac{(\textit{proof width/degree})^2}{N}\right)\right)$$

Yields superpolynomial bounds for width/degree $\omega\left(\sqrt{N \log N}\right)$
(and no implications for smaller width/degree [BG01, GL10])

Width/Degree Lower Bounds Imply Size Lower Bounds

Theorem ([IPS99, BW01])

For k-CNF formula over N variables

$$\textit{proof size} \geq \exp\left(\Omega\left(\frac{(\textit{proof width/degree})^2}{N}\right)\right)$$

Yields superpolynomial bounds for width/degree $\omega\left(\sqrt{N \log N}\right)$ (and no implications for smaller width/degree [BG01, GL10])

Resolution

- Well-developed machinery for width lower bounds
- One of many available tools

Polynomial calculus

- Degree lower bound machinery way less developed
- And pretty much only tool?!

Conversion to k-CNF "Graph Versions" of Formulas

- Need bounded width to use lower bound in [IPS99, BW01]
- But PHP formulas have wide clauses
- **Solution:** Restrict formulas to bounded-degree graphs

Conversion to k-CNF "Graph Versions" of Formulas

- Need bounded width to use lower bound in [IPS99, BW01]
- But PHP formulas have wide clauses
- **Solution:** Restrict formulas to bounded-degree graphs

For graph (onto functional) PHP, pigeons can fly only to neighbour holes:

$\bigvee_{j \in \mathcal{N}(i)} p_{i,j}$	pigeon i goes into hole in $\mathcal{N}(i)$
$\bigvee_{i \in \mathcal{N}(j)} p_{i,j}$	hole j gets pigeon from $\mathcal{N}(j)$

Conversion to k-CNF "Graph Versions" of Formulas

- Need bounded width to use lower bound in [IPS99, BW01]
- But PHP formulas have wide clauses
- **Solution:** Restrict formulas to bounded-degree graphs

For graph (onto functional) PHP, pigeons can fly only to neighbour holes:

$\bigvee_{j \in \mathcal{N}(i)} p_{i,j}$	pigeon i goes into hole in $\mathcal{N}(i)$
$\bigvee_{i \in \mathcal{N}(j)} p_{i,j}$	hole j gets pigeon from $\mathcal{N}(j)$

- Now strong width lower bounds $\Rightarrow$ strong size lower bounds
- And size lower bounds hold for original, unrestricted formulas
- Lower bounds for graph PHP also of independent interest

Lower Bounds via Graph Expansion

Standard approach:

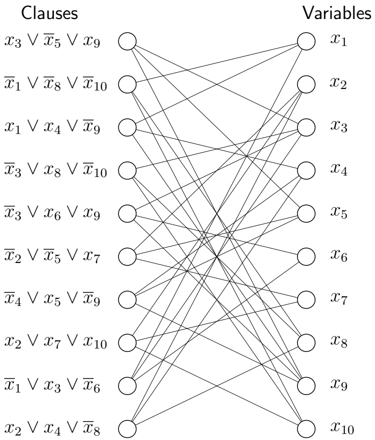
Lower bounds from expansion

Simplest example is the clause-variable incidence graph (CVIG)

Lower Bounds via Graph Expansion

Standard approach:
Lower bounds from expansion

Simplest example is the clause-variable incidence graph (CVIG)



Lower Bounds via Graph Expansion

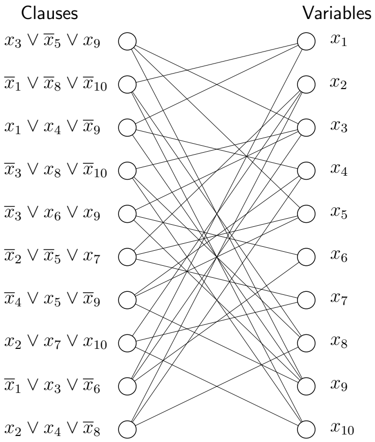
Standard approach:

Lower bounds from expansion

Simplest example is the clause-variable incidence graph (CVIG)

Boundary expansion:

Subsets of left vertices have many unique right neighbours



Lower Bounds via Graph Expansion

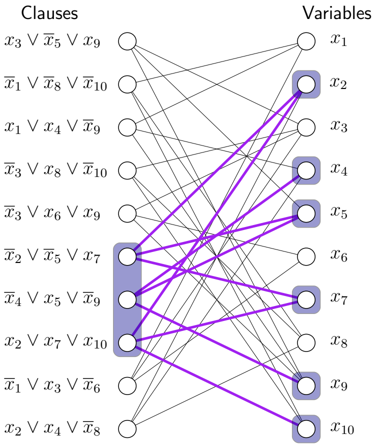
Standard approach:

Lower bounds from expansion

Simplest example is the clause-variable incidence graph (CVIG)

Boundary expansion:

Subsets of left vertices have many unique right neighbours



Lower Bounds via Graph Expansion

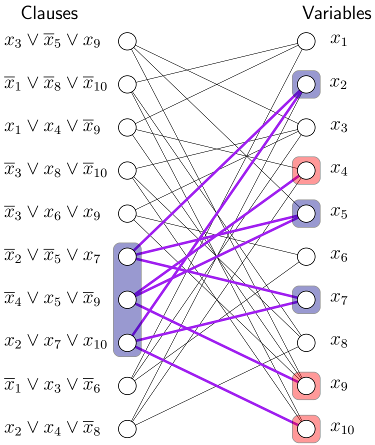
Standard approach:

Lower bounds from expansion

Simplest example is the clause-variable incidence graph (CVIG)

Boundary expansion:

Subsets of left vertices have many unique right neighbours

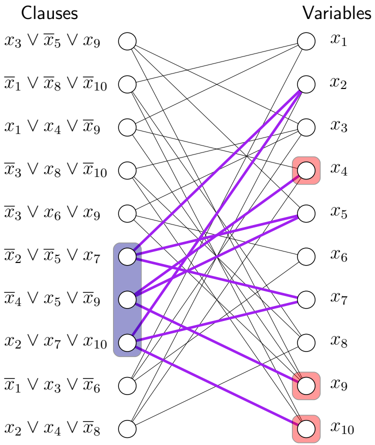


Lower Bounds via Graph Expansion

Standard approach:
Lower bounds from expansion

Simplest example is the clause-variable incidence graph (CVIG)

Boundary expansion:
Subsets of left vertices have many unique right neighbours



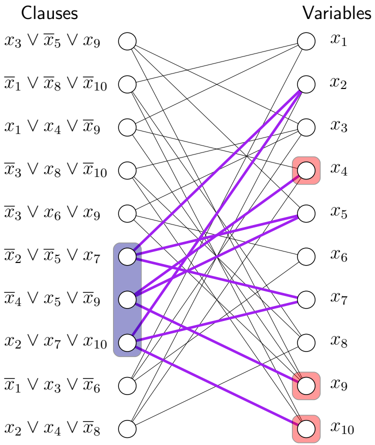
Lower Bounds via Graph Expansion

Standard approach:
Lower bounds from expansion

Simplest example is the clause-variable incidence graph (CVIG)

Boundary expansion:
Subsets of left vertices have many unique right neighbours

Problem:
CVIG often loses expansion of combinatorial problem



Lower Bounds via Graph Expansion

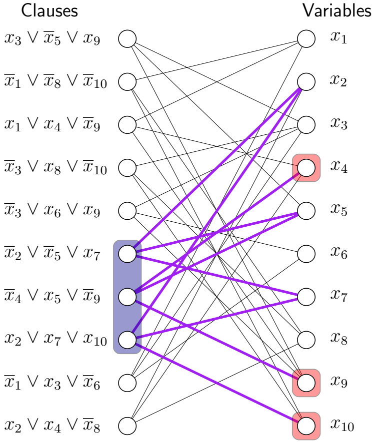
Standard approach:
Lower bounds from expansion

Simplest example is the clause-variable incidence graph (CVIG)

Boundary expansion:
Subsets of left vertices have many unique right neighbours

Problem:
CVIG often loses expansion of combinatorial problem

Need graph capturing combinatorial structure!



Generalized Incidence Graphs for CNF Formulas

Given CNF formula $\mathcal{F}$ over variables $\mathcal{V}$

- Partition clauses into $\mathcal{F} = E \cup \bigcup_{i=1}^{m} F_i$ (for E satisifiable)
- Divide variables into $\mathcal{V} = \bigcup_{j=1}^{n} V_j$ — **not** always partition
- Overlap ℓ: Any x appears in $\leq \ell$ different V_j

Generalized Incidence Graphs for CNF Formulas

Given CNF formula $\mathcal{F}$ over variables $\mathcal{V}$

- Partition clauses into $\mathcal{F} = E \cup \bigcup_{i=1}^{m} F_i$ (for E satisifiable)
- Divide variables into $\mathcal{V} = \bigcup_{j=1}^{n} V_j$ — **not** always partition
- Overlap ℓ: Any x appears in $\leq \ell$ different V_j

Build bipartite $(\mathcal{U}, \mathcal{V})_E$-graph $\mathcal{G}$

- Left vertices $\mathcal{U} = \{F_1, \ldots, F_m\}$
- Right vertices $\mathcal{V} = \{V_1, \ldots, V_n\}$
- Edge (F_i, V_j) if $\mathit{Vars}(F_i) \cap V_j \neq \emptyset$

Generalized Incidence Graphs for CNF Formulas

Given CNF formula $\mathcal{F}$ over variables $\mathcal{V}$

- Partition clauses into $\mathcal{F} = E \cup \bigcup_{i=1}^{m} F_i$ (for E satisifiable)
- Divide variables into $\mathcal{V} = \bigcup_{j=1}^{n} V_j$ — **not** always partition
- Overlap ℓ: Any x appears in $\leq \ell$ different V_j

Build bipartite $(\mathcal{U}, \mathcal{V})_E$-graph $\mathcal{G}$

- Left vertices $\mathcal{U} = \{F_1, \ldots, F_m\}$
- Right vertices $\mathcal{V} = \{V_1, \ldots, V_n\}$
- Edge (F_i, V_j) if $\mathit{Vars}(F_i) \cap V_j \neq \emptyset$

E not part of graph, but "filters" which assignments to consider
(E.g., partial matchings for pigeonhole principle formulas)

The Resolution Edge Game

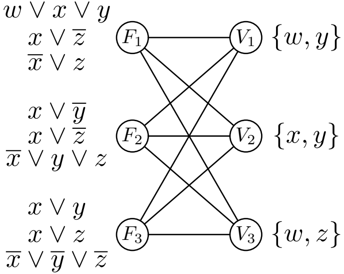
Resolution edge game on (F_i, V_j) w.r.t. "filtering set" E

- Adversary choses any total assignment α such that $\alpha(E) = 1$
- We can modify α on V_j to get α'
- We win if $\alpha'(F_i \wedge E) = 1$

The Resolution Edge Game

Resolution edge game on (F_i, V_j) w.r.t. "filtering set" E

- Adversary choses any total assignment α such that $\alpha(E) = 1$
- We can modify α on V_j to get α'
- We win if $\alpha'(F_i \wedge E) = 1$

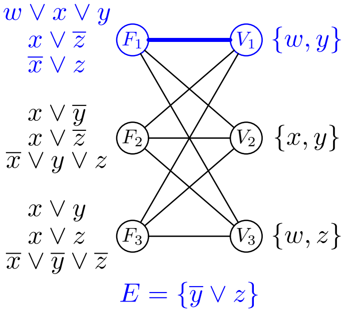


$E = \{\overline{y} \vee z\}$

The Resolution Edge Game

Resolution edge game on (F_i, V_j) w.r.t. "filtering set" E

- Adversary choses any total assignment α such that $\alpha(E) = 1$
- We can modify α on V_j to get α'
- We win if $\alpha'(F_i \wedge E) = 1$

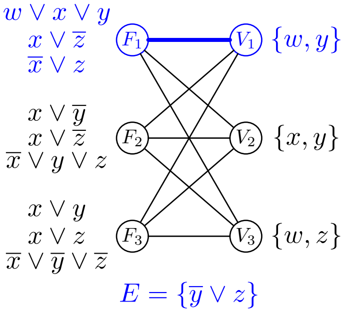


Edge game on (F_1, V_1) w.r.t. E

The Resolution Edge Game

Resolution edge game on (F_i, V_j) w.r.t. "filtering set" E

- Adversary choses any total assignment α such that $\alpha(E) = 1$
- We can modify α on V_j to get α'
- We win if $\alpha'(F_i \wedge E) = 1$



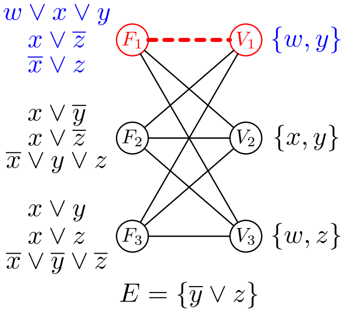
Edge game on (F_1, V_1) w.r.t. E

Take $\alpha_1 = \{w = y = z = 0, x = 1\}$

The Resolution Edge Game

Resolution edge game on (F_i, V_j) w.r.t. "filtering set" E

- Adversary choses any total assignment α such that $\alpha(E) = 1$
- We can modify α on V_j to get α'
- We win if $\alpha'(F_i \wedge E) = 1$



Edge game on (F_1, V_1) w.r.t. E

Take $\alpha_1 = \{w = y = z = 0, x = 1\}$

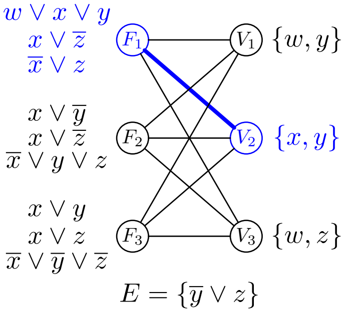
Can't win, since

- $\alpha_1(\overline{x} \vee z) = 0$
- can't flip x or z (not in V_1)

The Resolution Edge Game

Resolution edge game on (F_i, V_j) w.r.t. "filtering set" E

- Adversary choses any total assignment α such that $\alpha(E) = 1$
- We can modify α on V_j to get α'
- We win if $\alpha'(F_i \wedge E) = 1$

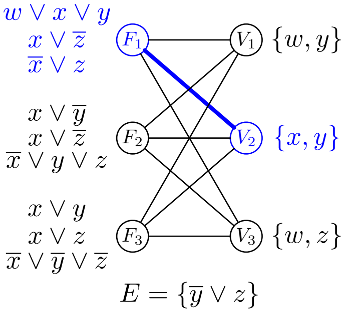


Edge game on (F_1, V_2) w.r.t. E

The Resolution Edge Game

Resolution edge game on (F_i, V_j) w.r.t. "filtering set" E

- Adversary choses any total assignment α such that $\alpha(E) = 1$
- We can modify α on V_j to get α'
- We win if $\alpha'(F_i \wedge E) = 1$



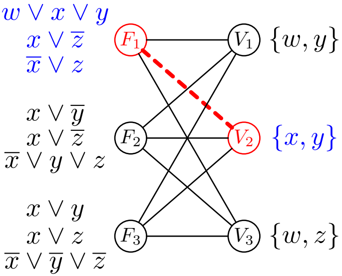
Edge game on (F_1, V_2) w.r.t. E

Take $\alpha_2 = \{w = y = z = 0, x = *\}$

The Resolution Edge Game

Resolution edge game on (F_i, V_j) w.r.t. "filtering set" E

- Adversary choses any total assignment α such that $\alpha(E) = 1$
- We can modify α on V_j to get α'
- We win if $\alpha'(F_i \land E) = 1$



$E = \{\overline{y} \lor z\}$

Edge game on (F_1, V_2) w.r.t. E

Take $\alpha_2 = \{w = y = z = 0, x = *\}$

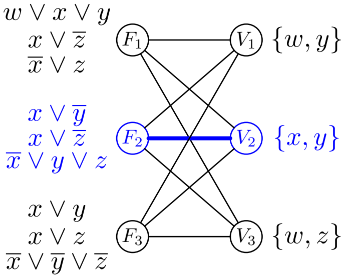
Again can't win, since

- can't flip w or z (not in V_2)
- flipping $y \in V_2$ falsifies E
- $F_1{\restriction_{\{w=y=z=0\}}} = \{x, \overline{x}\}$

The Resolution Edge Game

Resolution edge game on (F_i, V_j) w.r.t. "filtering set" E

- Adversary choses any total assignment α such that $\alpha(E) = 1$
- We can modify α on V_j to get α'
- We win if $\alpha'(F_i \wedge E) = 1$



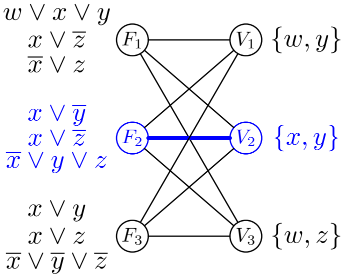
$E = \{\overline{y} \vee z\}$

Edge game on (F_2, V_2) w.r.t. E

The Resolution Edge Game

Resolution edge game on (F_i, V_j) w.r.t. "filtering set" E

- Adversary choses any total assignment α such that $\alpha(E) = 1$
- We can modify α on V_j to get α'
- We win if $\alpha'(F_i \wedge E) = 1$



$E = \{\overline{y} \vee z\}$

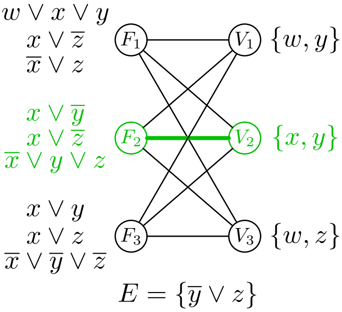
Edge game on (F_2, V_2) w.r.t. E

Now we can win!

The Resolution Edge Game

Resolution edge game on (F_i, V_j) w.r.t. "filtering set" E

- Adversary choses any total assignment α such that $\alpha(E) = 1$
- We can modify α on V_j to get α'
- We win if $\alpha'(F_i \wedge E) = 1$



Edge game on (F_2, V_2) w.r.t. E

Now we can win!

Given any α_3 s.t. $\alpha_3(E) = 1$:

- assign $\alpha'(x) = \alpha_3(y \vee z)$
- E still OK — didn't touch y, z
- F_2 OK — encodes $x \leftrightarrow (y \vee z)$

Edge Game, Expansion, and Width Lower Bounds

Recall boundary $\partial(\mathcal{U}') = \{V \in \mathcal{N}(\mathcal{U}') \mid \mathcal{N}(V) \cap \mathcal{U}' = \{F\} \text{ unique}\}$

Edge Game, Expansion, and Width Lower Bounds

Recall boundary $\partial(\mathcal{U}') = \{V \in \mathcal{N}(\mathcal{U}') \mid \mathcal{N}(V) \cap \mathcal{U}' = \{F\} \text{ unique}\}$

Resolution expander

Say that an $(\mathcal{U}, \mathcal{V})_E$-graph is an (s, δ, E)-resolution expander if

- For all $\mathcal{U}' \subseteq \mathcal{U}$, $|\mathcal{U}'| \leq s$ it holds that $|\partial(\mathcal{U}')| \geq \delta|\mathcal{U}'|$
- For all edges (F_i, V_j) we can win the resolution edge game with respect to E

Edge Game, Expansion, and Width Lower Bounds

Recall boundary $\partial(\mathcal{U}') = \{V \in \mathcal{N}(\mathcal{U}') \mid \mathcal{N}(V) \cap \mathcal{U}' = \{F\} \text{ unique}\}$

Resolution expander

Say that an $(\mathcal{U}, \mathcal{V})_E$-graph is an (s, δ, E)-resolution expander if

- For all $\mathcal{U}' \subseteq \mathcal{U}$, $|\mathcal{U}'| \leq s$ it holds that $|\partial(\mathcal{U}')| \geq \delta |\mathcal{U}'|$
- For all edges (F_i, V_j) we can win the resolution edge game with respect to E

Theorem (essentially [BW01])

If the CNF formula $\mathcal{F}$ admits an (s, δ, E)-resolution expander with overlap ℓ, then

$$\textit{resolution proof width} > \frac{\delta s}{2\ell}$$

Ben-Sasson–Wigderson à la Alekhnovich–Razborov

Theorem (essentially [BW01])

If $\mathcal{F}$ admits an (s, δ, E)-resolution expander with overlap ℓ, then

$$\textit{resolution proof width} > \frac{\delta s}{2\ell}$$

Proof sketch (in the style of [AR03]):

Let $\pi = (C_1, C_2, C_3, \ldots)$ be derivation from $\mathcal{F}$ in width $\leq \frac{\delta s}{2\ell}$

Ben-Sasson–Wigderson à la Alekhnovich–Razborov

Theorem (essentially [BW01])

If $\mathcal{F}$ admits an (s, δ, E)-resolution expander with overlap ℓ, then

$$\textit{resolution proof width} > \frac{\delta s}{2\ell}$$

Proof sketch (in the style of [AR03]):

Let $\pi = (C_1, C_2, C_3, \ldots)$ be derivation from $\mathcal{F}$ in width $\leq \frac{\delta s}{2\ell}$

For every $C_i \in \pi$, define "support" $Sup_s(C_i) \subseteq \mathcal{F} \setminus E$ such that

1. $|Sup_s(C_i)| \leq s/2$
2. $Sup_s(C_i) \cup E \vDash C_i$

Ben-Sasson–Wigderson à la Alekhnovich–Razborov

Theorem (essentially [BW01])

If $\mathcal{F}$ admits an (s, δ, E)-resolution expander with overlap ℓ, then

$$\textit{resolution proof width} > \frac{\delta s}{2\ell}$$

Proof sketch (in the style of [AR03]):

Let $\pi = (C_1, C_2, C_3, \ldots)$ be derivation from $\mathcal{F}$ in width $\leq \frac{\delta s}{2\ell}$

For every $C_i \in \pi$, define "support" $Sup_s(C_i) \subseteq \mathcal{F} \setminus E$ such that

1. $|Sup_s(C_i)| \leq s/2$
2. $Sup_s(C_i) \cup E \vDash C_i$

$\Rightarrow$ $\left|Sup_s(C_i)\right|$ so small that $Sup_s(C_i) \cup E$ satisfiable

$\Rightarrow$ $Sup_s(C_i) \cup E \vDash C_i$ means that C_i satisfiable (hence not $\perp$) $\square$

Support

Clause neighbourhood $\mathcal{N}(C) = \{V \in \mathcal{V} \mid \mathit{Vars}(C) \cap V \neq \emptyset\}$

Left-side set $\mathcal{U}' \subseteq \mathcal{U}$ in $(\mathcal{U}, \mathcal{V})_E$-graph is (s, C)-contained if

- $|\mathcal{U}'| \leq s$
- $\partial(\mathcal{U}') \subseteq \mathcal{N}(C)$

s-support $\mathit{Sup}_s(C)$ of C = largest (s, C)-contained subset
(Intuition: "largest clause set possibly used to derive C")

Support

Clause neighbourhood $\mathcal{N}(C) = \{V \in \mathcal{V} \mid Vars(C) \cap V \neq \emptyset\}$

Left-side set $\mathcal{U}' \subseteq \mathcal{U}$ in $(\mathcal{U}, \mathcal{V})_E$-graph is (s, C)-contained if

- $|\mathcal{U}'| \leq s$
- $\partial(\mathcal{U}') \subseteq \mathcal{N}(C)$

s-support $Sup_s(C)$ of C = largest (s, C)-contained subset
(Intuition: "largest clause set possibly used to derive C")

Need to argue:

- $Sup_s(C_i)$ well-defined — by expansion
- $|Sup_s(C_i)| \leq s/2$ — also by expansion
- $Sup_s(C_i) \cup E \vDash C_i$ — by resolution edge game and induction

Applications: Tseitin and Onto-FPHP

Tseitin formulas

- F_i = clauses encoding parity constraint for ith vertex
- V_j = singleton set with jth edge (so overlap $\ell = 1$)
- $E = \emptyset$
- If underlying graph edge expander, then $(\mathcal{U}, \mathcal{V})_E$-graph is resolution expander with same parameters

Applications: Tseitin and Onto-FPHP

Tseitin formulas

- F_i = clauses encoding parity constraint for ith vertex
- V_j = singleton set with jth edge (so overlap $\ell = 1$)
- $E = \emptyset$
- If underlying graph edge expander, then $(\mathcal{U}, \mathcal{V})_E$-graph is resolution expander with same parameters

Onto functional PHP formulas

- F_i = singleton set with pigeon axiom for pigeon i
- V_j = all variables $p_{i,j}$ mentioning hole j (again overlap $\ell = 1$)
- E = all hole, functional, and onto axioms
- If onto FPHP restricted to bipartite graph, then $(\mathcal{U}, \mathcal{V})_E$-graph is resolution expander with same parameters

From Resolution to Polynomial Calculus

So far: Obtain resolution width lower bounds from expander graphs where we can win following game on all edges

Resolution edge game on (F, V) with respect to E

1. Adversary provides total assignment α such that $\alpha(E) = 1$
2. Choose $\rho_V : V \to \{0, 1\}$ so that $\alpha[\rho_V/V](F \wedge E) = 1$

From Resolution to Polynomial Calculus

So far: Obtain resolution width lower bounds from expander graphs where we can win following game on all edges

Resolution edge game on (F, V) with respect to E

1. Adversary provides total assignment α such that $\alpha(E) = 1$
2. Choose $\rho_V : V \to \{0, 1\}$ so that $\alpha[\rho_V / V](F \land E) = 1$

But Tseitin and onto FPHP both easy for polynomial calculus!

From Resolution to Polynomial Calculus

So far: Obtain resolution width lower bounds from expander graphs where we can win following game on all edges

Resolution edge game on (F, V) with respect to E

1. Adversary provides total assignment α such that $\alpha(E) = 1$
2. Choose $\rho_V : V \to \{0, 1\}$ so that $\alpha[\rho_V/V](F \wedge E) = 1$

But Tseitin and onto FPHP both easy for polynomial calculus!

Polynomial calculus degree lower bounds require harder game

Polynomial calculus edge game on (F, V) with respect to E

1. Commit to partial assignment $\rho_V : V \to \{0, 1\}$
2. Adversary provides total assignment α such that $\alpha(E) = 1$
3. Substituting ρ_V for V should yield $\alpha[\rho_V/V](F \wedge E) = 1$

The Polynomial Calculus Edge Game

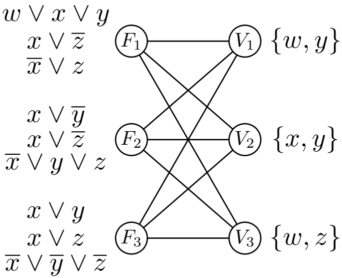
To win PC edge game on (F, V), need to find $\rho_V : V \to \{0, 1\}$ s.t.

- $\rho_V(F) = 1$
- $\rho_V(C) = 1$ for all clauses $C \in E$ with $V \cap \mathit{Vars}(C) \neq \emptyset$

The Polynomial Calculus Edge Game

To win PC edge game on (F, V), need to find $\rho_V : V \to \{0,1\}$ s.t.

- $\rho_V(F) = 1$
- $\rho_V(C) = 1$ for all clauses $C \in E$ with $V \cap Vars(C) \neq \emptyset$

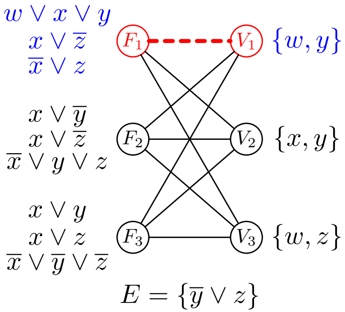


$E = \{\overline{y} \lor z\}$

The Polynomial Calculus Edge Game

To win PC edge game on (F, V), need to find $\rho_V : V \to \{0,1\}$ s.t.

- $\rho_V(F) = 1$
- $\rho_V(C) = 1$ for all clauses $C \in E$ with $V \cap Vars(C) \neq \emptyset$



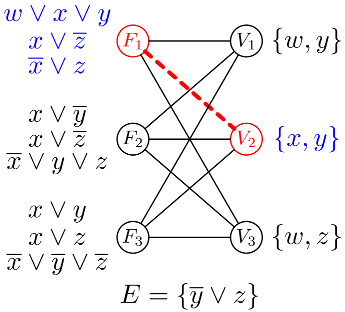
Recall that for resolution edge game we:

- Lose on (F_1, V_1)

The Polynomial Calculus Edge Game

To win PC edge game on (F, V), need to find $\rho_V : V \to \{0, 1\}$ s.t.

- $\rho_V(F) = 1$
- $\rho_V(C) = 1$ for all clauses $C \in E$ with $V \cap Vars(C) \neq \emptyset$



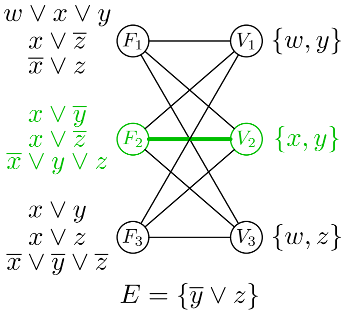
Recall that for resolution edge game we:

- Lose on (F_1, V_1)
- Lose on (F_1, V_2)

The Polynomial Calculus Edge Game

To win PC edge game on (F, V), need to find $\rho_V : V \to \{0,1\}$ s.t.

- $\rho_V(F) = 1$
- $\rho_V(C) = 1$ for all clauses $C \in E$ with $V \cap Vars(C) \neq \emptyset$



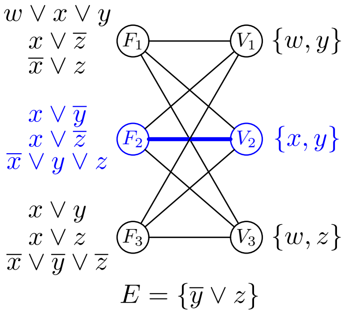
Recall that for resolution edge game we:

- Lose on (F_1, V_1)
- Lose on (F_1, V_2)
- Win on (F_2, V_2)

The Polynomial Calculus Edge Game

To win PC edge game on (F, V), need to find $\rho_V : V \to \{0,1\}$ s.t.

- $\rho_V(F) = 1$
- $\rho_V(C) = 1$ for all clauses $C \in E$ with $V \cap Vars(C) \neq \emptyset$

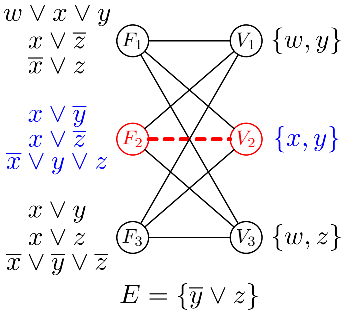


PC edge game on (F_2, V_2) w.r.t. E

The Polynomial Calculus Edge Game

To win PC edge game on (F, V), need to find $\rho_V : V \rightarrow \{0, 1\}$ s.t.

- $\rho_V(F) = 1$
- $\rho_V(C) = 1$ for all clauses $C \in E$ with $V \cap Vars(C) \neq \emptyset$



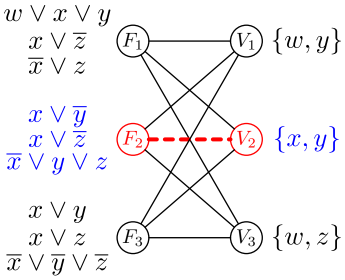
PC edge game on (F_2, V_2) w.r.t. E

Now we can't win

The Polynomial Calculus Edge Game

To win PC edge game on (F, V), need to find $\rho_V : V \to \{0,1\}$ s.t.

- $\rho_V(F) = 1$
- $\rho_V(C) = 1$ for all clauses $C \in E$ with $V \cap Vars(C) \neq \emptyset$



$E = \{\overline{y} \lor z\}$

PC edge game on (F_2, V_2) w.r.t. E

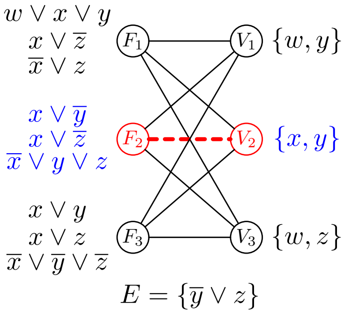
Now we can't win

- $E = \{\overline{y} \lor z\}$ needs $\rho_V(y) = 0$

The Polynomial Calculus Edge Game

To win PC edge game on (F, V), need to find $\rho_V : V \to \{0, 1\}$ s.t.

- $\rho_V(F) = 1$
- $\rho_V(C) = 1$ for all clauses $C \in E$ with $V \cap Vars(C) \neq \emptyset$



PC edge game on (F_2, V_2) w.r.t. E

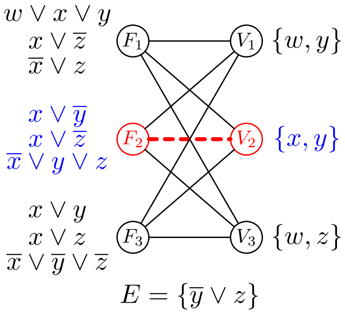
Now we can't win

- $E = \{\overline{y} \lor z\}$ needs $\rho_V(y) = 0$
- But $F_2\restriction_{\{y=0\}} = \{x \lor \overline{z}, \overline{x} \lor z\}$

The Polynomial Calculus Edge Game

To win PC edge game on (F, V), need to find $\rho_V : V \to \{0, 1\}$ s.t.

- $\rho_V(F) = 1$
- $\rho_V(C) = 1$ for all clauses $C \in E$ with $V \cap \mathit{Vars}(C) \neq \emptyset$



PC edge game on (F_2, V_2) w.r.t. E

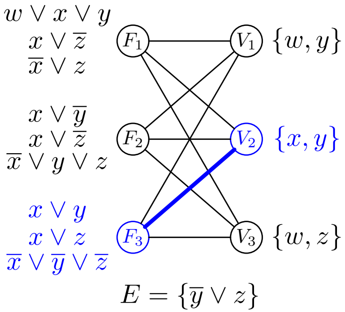
Now we can't win

- $E = \{\overline{y} \lor z\}$ needs $\rho_V(y) = 0$
- But $F_2\restriction_{\{y=0\}} = \{x \lor \overline{z}, \overline{x} \lor z\}$
- Adversary sets $\alpha_V(z) = 1 - \rho_V(x)$

The Polynomial Calculus Edge Game

To win PC edge game on (F, V), need to find $\rho_V : V \to \{0, 1\}$ s.t.

- $\rho_V(F) = 1$
- $\rho_V(C) = 1$ for all clauses $C \in E$ with $V \cap Vars(C) \neq \emptyset$

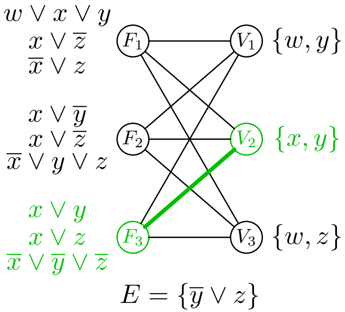


PC edge game on (F_3, V_2) w.r.t. E

The Polynomial Calculus Edge Game

To win PC edge game on (F, V), need to find $\rho_V : V \to \{0,1\}$ s.t.

- $\rho_V(F) = 1$
- $\rho_V(C) = 1$ for all clauses $C \in E$ with $V \cap Vars(C) \neq \emptyset$



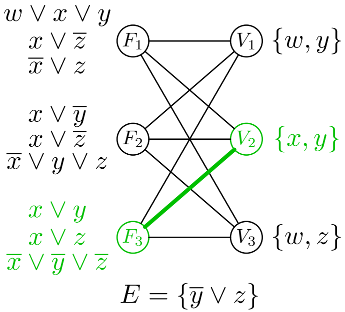
PC edge game on (F_3, V_2) w.r.t. E

On this edge we can win!

The Polynomial Calculus Edge Game

To win PC edge game on (F, V), need to find $\rho_V : V \to \{0,1\}$ s.t.

- $\rho_V(F) = 1$
- $\rho_V(C) = 1$ for all clauses $C \in E$ with $V \cap Vars(C) \neq \emptyset$



PC edge game on (F_3, V_2) w.r.t. E

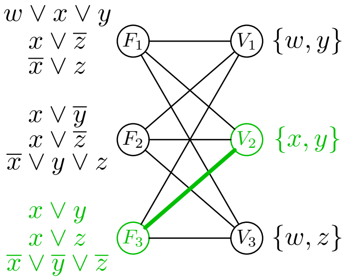
On this edge we can win!

- Choose $\rho_V = \{x = 1, y = 0\}$

The Polynomial Calculus Edge Game

To win PC edge game on (F, V), need to find $\rho_V : V \to \{0,1\}$ s.t.

- $\rho_V(F) = 1$
- $\rho_V(C) = 1$ for all clauses $C \in E$ with $V \cap Vars(C) \neq \emptyset$



$E = \{\overline{y} \lor z\}$

PC edge game on (F_3, V_2) w.r.t. E

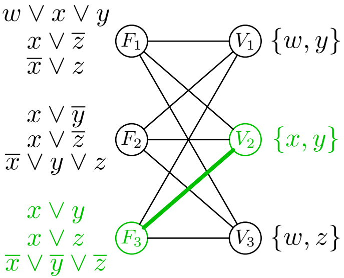
On this edge we can win!

- Choose $\rho_V = \{x = 1, y = 0\}$
- $\rho_V(F_3) = 1$

The Polynomial Calculus Edge Game

To win PC edge game on (F, V), need to find $\rho_V : V \to \{0, 1\}$ s.t.

- $\rho_V(F) = 1$
- $\rho_V(C) = 1$ for all clauses $C \in E$ with $V \cap Vars(C) \neq \emptyset$



$E = \{\overline{y} \lor z\}$

PC edge game on (F_3, V_2) w.r.t. E

On this edge we can win!

- Choose $\rho_V = \{x = 1, y = 0\}$
- $\rho_V(F_3) = 1$
- $\rho_V(E) = 1$

A Generalized Method for PC Degree Lower Bounds

Polynomial calculus expander

Say that an $(\mathcal{U}, \mathcal{V})_E$-graph is an (s, δ, E)-PC expander if

- For all $\mathcal{U}' \subseteq \mathcal{U}$, $|\mathcal{U}'| \leq s$ it holds that $|\partial(\mathcal{U}')| \geq \delta|\mathcal{U}'|$
- For all edges (F_i, V_j) we can win the PC edge game with respect to E

A Generalized Method for PC Degree Lower Bounds

Polynomial calculus expander

Say that an $(\mathcal{U}, \mathcal{V})_E$-graph is an (s, δ, E)-PC expander if

- For all $\mathcal{U}' \subseteq \mathcal{U}$, $|\mathcal{U}'| \leq s$ it holds that $|\partial(\mathcal{U}')| \geq \delta|\mathcal{U}'|$
- For all edges (F_i, V_j) we can win the PC edge game with respect to E

Theorem ([MN15] building on [AR03])

If $\mathcal{F}$ admits an (s, δ, E)-PC expander with overlap ℓ, then

$$\textit{PC proof degree} > \frac{\delta s}{2\ell}$$

A Generalized Method for PC Degree Lower Bounds

Polynomial calculus expander

Say that an $(\mathcal{U}, \mathcal{V})_E$-graph is an (s, δ, E)-PC expander if

- For all $\mathcal{U}' \subseteq \mathcal{U}$, $|\mathcal{U}'| \leq s$ it holds that $|\partial(\mathcal{U}')| \geq \delta|\mathcal{U}'|$
- For all edges (F_i, V_j) we can win the PC edge game with respect to E

Theorem ([MN15] building on [AR03])

If $\mathcal{F}$ admits an (s, δ, E)-PC expander with overlap ℓ, then

$$\textit{PC proof degree} > \frac{\delta s}{2\ell}$$

Also holds for sets of polynomials not obtained from CNFs
Proof by carefully adapting [AR03] (fairly involved — can't say much)

Consequences

Common framework for previous lower bounds

- Random k-CNF formulas [AR03]
- CNF formulas with expanding CVIGs [AR03]
- "Vanilla" PHP formulas [AR03]
- Ordering principle formulas [GL10]
- Subset cardinality formulas [MN14]

Consequences

Common framework for previous lower bounds

- Random k-CNF formulas [AR03]
- CNF formulas with expanding CVIGs [AR03]
- "Vanilla" PHP formulas [AR03]
- Ordering principle formulas [GL10]
- Subset cardinality formulas [MN14]

New lower bounds

- Functional pigeonhole principle [MN15]
- Graph colouring [LN17]

Hardness of Different Flavours of PHP

Variant	Resolution	Polynomial calculus
PHP		
FPHP		
Onto-PHP		
Onto-FPHP		

Hardness of Different Flavours of PHP

Variant	Resolution	Polynomial calculus
PHP	hard [Hak85]	
FPHP		
Onto-PHP		
Onto-FPHP		

Hardness of Different Flavours of PHP

Variant	Resolution	Polynomial calculus
PHP	hard [Hak85]	
FPHP	hard [Hak85]	
Onto-PHP	hard [Hak85]	
Onto-FPHP	hard [Hak85]	

Hardness of Different Flavours of PHP

Variant	Resolution	Polynomial calculus
PHP	hard [Hak85]	hard [AR03]
FPHP	hard [Hak85]	
Onto-PHP	hard [Hak85]	
Onto-FPHP	hard [Hak85]	

Hardness of Different Flavours of PHP

Variant	Resolution	Polynomial calculus
PHP	hard [Hak85]	hard [AR03]
FPHP	hard [Hak85]	
Onto-PHP	hard [Hak85]	
Onto-FPHP	hard [Hak85]	easy! [Rii93]

Hardness of Different Flavours of PHP

Variant	Resolution	Polynomial calculus
PHP	hard [Hak85]	hard [AR03]
FPHP	hard [Hak85]	?
Onto-PHP	hard [Hak85]	?
Onto-FPHP	hard [Hak85]	easy! [Rii93]

Hardness of Different Flavours of PHP

Variant	Resolution	Polynomial calculus
PHP	hard [Hak85]	hard [AR03]
FPHP	hard [Hak85]	?
Onto-PHP	hard [Hak85]	hard [AR03]
Onto-FPHP	hard [Hak85]	easy! [Rii93]

Joint work with Mladen Mikša [MN15]:

- Observe that [AR03] proves hardness of Onto-PHP

Hardness of Different Flavours of PHP

Variant	Resolution	Polynomial calculus
PHP	hard [Hak85]	hard [AR03]
FPHP	hard [Hak85]	hard [MN15]
Onto-PHP	hard [Hak85]	hard [AR03]
Onto-FPHP	hard [Hak85]	easy! [Rii93]

Joint work with Mladen Mikša [MN15]:

- Observe that [AR03] proves hardness of Onto-PHP
- Prove that functional PHP is hard for polynomial calculus (answering open question in [Raz02, Raz14])

Degree Lower Bound for Functional PHP

Theorem ([MN15])

If G is a (standard) bipartite (s, δ)-boundary expander with left degree $\leq d$, then $FPHP_G$ requires PC degree $> \delta s/(2d)$

Degree Lower Bound for Functional PHP

Theorem ([MN15])

If G is a (standard) bipartite (s, δ)-boundary expander with left degree $\leq d$, then $FPHP_G$ requires PC degree $> \delta s/(2d)$

Proof: Just need to build expanding $(\mathcal{U}, \mathcal{V})_E$-graph

Degree Lower Bound for Functional PHP

Theorem ([MN15])

If G is a (standard) bipartite (s, δ)-boundary expander with left degree $\leq d$, then $FPHP_G$ requires PC degree $> \delta s/(2d)$

Proof: Just need to build expanding $(\mathcal{U}, \mathcal{V})_E$-graph

- F_i = pigeon axiom for pigeon i

Degree Lower Bound for Functional PHP

Theorem ([MN15])

If G is a (standard) bipartite (s, δ)-boundary expander with left degree $\leq d$, then $FPHP_G$ requires PC degree $> \delta s/(2d)$

Proof: Just need to build expanding $(\mathcal{U}, \mathcal{V})_E$-graph

- F_i = pigeon axiom for pigeon i
- E = all hole and functional axioms

Degree Lower Bound for Functional PHP

Theorem ([MN15])

If G is a (standard) bipartite (s, δ)-boundary expander with left degree $\leq d$, then $FPHP_G$ requires PC degree $> \delta s/(2d)$

Proof: Just need to build expanding $(\mathcal{U}, \mathcal{V})_E$-graph

- $F_i =$ pigeon axiom for pigeon i
- $E =$ all hole and functional axioms
- $V_j = \{p_{i',j'} \mid i' \in \mathcal{N}(j) \text{ and } j' \in \mathcal{N}(i')\}$
  "All holes pigeons incident to hole j can go to"

Degree Lower Bound for Functional PHP

Theorem ([MN15])

If G is a (standard) bipartite (s, δ)-boundary expander with left degree $\leq d$, then $FPHP_G$ requires PC degree $> \delta s/(2d)$

Proof: Just need to build expanding $(\mathcal{U}, \mathcal{V})_E$-graph

- F_i = pigeon axiom for pigeon i
- E = all hole and functional axioms
- $V_j = \{p_{i',j'} \mid i' \in \mathcal{N}(j)$ and $j' \in \mathcal{N}(i')\}$
  "All holes pigeons incident to hole j can go to"
- Can prove (straightforward exercise):
  - Overlap ℓ satisfies $1 < \ell \leq d$
  - Can win PC edge game on all edges (F_i, V_j)
  - Original graph G and $(\mathcal{U}, \mathcal{V})_E$ are isomorphic

Degree Lower Bound for Functional PHP

Theorem ([MN15])

If G is a (standard) bipartite (s, δ)-boundary expander with left degree $\leq d$, then $FPHP_G$ requires PC degree $> \delta s/(2d)$

Proof: Just need to build expanding $(\mathcal{U}, \mathcal{V})_E$-graph

- F_i = pigeon axiom for pigeon i
- E = all hole and functional axioms
- $V_j = \left\{p_{i',j'} \,\middle|\, i' \in \mathcal{N}(j) \text{ and } j' \in \mathcal{N}(i')\right\}$
  "All holes pigeons incident to hole j can go to"
- Can prove (straightforward exercise):
  - Overlap ℓ satisfies $1 < \ell \leq d$
  - Can win PC edge game on all edges (F_i, V_j)
  - Original graph G and $(\mathcal{U}, \mathcal{V})_E$ are isomorphic
- So get same expansion parameters, and theorem follows □

Graph Colouring

Graph k-colouring formulas

"$G = (V, E)$ is k-colourable"

Variables $x_{v,c}$ = "vertex v gets colour c"

$x_{v,1} \lor x_{v,2} \lor \cdots \lor x_{v,k}$	every vertex v gets a colour
$\overline{x}_{v,c} \lor \overline{x}_{v,c'}$	every vertex v is uniquely coloured
$\overline{x}_{u,c} \lor \overline{x}_{v,c}$	neighbours $(u, v) \in E$ get different colours

Graph Colouring

Graph k-colouring formulas

"$G = (V, E)$ is k-colourable"

Variables $x_{v,c}$ = "vertex v gets colour c"

$x_{v,1} \lor x_{v,2} \lor \cdots \lor x_{v,k}$	every vertex v gets a colour
$\overline{x}_{v,c} \lor \overline{x}_{v,c'}$	every vertex v is uniquely coloured
$\overline{x}_{u,c} \lor \overline{x}_{v,c}$	neighbours $(u, v) \in E$ get different colours

Average-case exponential lower bounds for resolution [BCMM05]

Graph Colouring

Graph k-colouring formulas

"$G = (V, E)$ is k-colourable"

Variables $x_{v,c}$ = "vertex v gets colour c"

$x_{v,1} \lor x_{v,2} \lor \cdots \lor x_{v,k}$	every vertex v gets a colour
$\overline{x}_{v,c} \lor \overline{x}_{v,c'}$	every vertex v is uniquely coloured
$\overline{x}_{u,c} \lor \overline{x}_{v,c}$	neighbours $(u, v) \in E$ get different colours

Average-case exponential lower bounds for resolution [BCMM05]

No lower bounds for polynomial calculus

Graph Colouring

Graph k-colouring formulas
"$G = (V, E)$ is k-colourable"

Variables $x_{v,c}$ = "vertex v gets colour c"

$x_{v,1} \lor x_{v,2} \lor \cdots \lor x_{v,k}$	every vertex v gets a colour
$\overline{x}_{v,c} \lor \overline{x}_{v,c'}$	every vertex v is uniquely coloured
$\overline{x}_{u,c} \lor \overline{x}_{v,c}$	neighbours $(u, v) \in E$ get different colours

Average-case exponential lower bounds for resolution [BCMM05]

No lower bounds for polynomial calculus

On the contrary, [DLMM08, DLMO09, DLMM11, DMP$^+$15] claim very efficient algorithms based on Nullstellensatz ("static PC") for slightly different encoding using primitive kth roots of unity

Polynomial Calculus Lower Bound for Colouring

Joint work with Massimo Lauria [LN17]:

Theorem ([LN17])

For any $k \geq 3$ $\exists$ constant-degree graphs which require linear PC degree, and hence exponential size, to be proven non-k-colourable

Polynomial Calculus Lower Bound for Colouring

Joint work with Massimo Lauria [LN17]:

Theorem ([LN17])

For any $k \geq 3$ $\exists$ constant-degree graphs which require linear PC degree, and hence exponential size, to be proven non-k-colourable

Proof idea:

- Reduce functional PHP instance to graph colouring instance
- Show that polynomial calculus "can compute this reduction"
- Hence these graph colouring instances must be hard

Polynomial Calculus Lower Bound for Colouring

Joint work with Massimo Lauria [LN17]:

Theorem ([LN17])

For any $k \geq 3$ $\exists$ constant-degree graphs which require linear PC degree, and hence exponential size, to be proven non-k-colourable

Proof idea:

- Reduce functional PHP instance to graph colouring instance
- Show that polynomial calculus "can compute this reduction"
- Hence these graph colouring instances must be hard

Lower bound applies also to kth-root-of-unity encoding

Answers open question raised in [DLMO09, LLO16]

Sketch of Reduction

- Given FPHP instance for bipartite graph of left degree k

Sketch of Reduction

- Given FPHP instance for bipartite graph of left degree k
- Order available holes $\mathcal{N}(i) = \{j_{i,1}, \ldots, j_{i,c}\}$ for every pigeon i

Sketch of Reduction

- Given FPHP instance for bipartite graph of left degree k
- Order available holes $\mathcal{N}(i) = \{j_{i,1}, \ldots, j_{i,c}\}$ for every pigeon i
- Vertex i coloured with colour c $\Leftrightarrow$ pigeon i flies to hole $j_{i,c}$

Sketch of Reduction

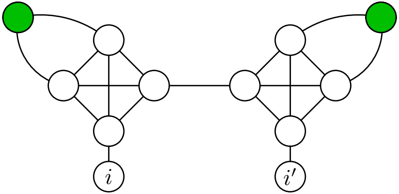
- Given FPHP instance for bipartite graph of left degree k
- Order available holes $\mathcal{N}(i) = \{j_{i,1}, \ldots, j_{i,c}\}$ for every pigeon i
- Vertex i coloured with colour c $\Leftrightarrow$ pigeon i flies to hole $j_{i,c}$
- $j_{i,c} = j_{i',c'}$ $\Rightarrow$ can't colour i by c and i' by c' simultaneously

Sketch of Reduction

- Given FPHP instance for bipartite graph of left degree k
- Order available holes $\mathcal{N}(i) = \{j_{i,1}, \ldots, j_{i,c}\}$ for every pigeon i
- Vertex i coloured with colour c $\Leftrightarrow$ pigeon i flies to hole $j_{i,c}$
- $j_{i,c} = j_{i',c'}$ $\Rightarrow$ can't colour i by c and i' by c' simultaneously
- *Almost* colouring, except forbidding specific colour pair (c, c') instead of arbitrary but same colour — fix with gadgets!

Sketch of Reduction

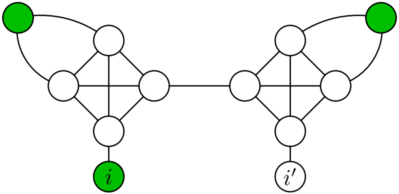
- Given FPHP instance for bipartite graph of left degree k
- Order available holes $\mathcal{N}(i) = \{j_{i,1}, \ldots, j_{i,c}\}$ for every pigeon i
- Vertex i coloured with colour $c \Leftrightarrow$ pigeon i flies to hole $j_{i,c}$
- $j_{i,c} = j_{i',c'} \Rightarrow$ can't colour i by c and i' by c' simultaneously
- *Almost* colouring, except forbidding specific colour pair (c, c') instead of arbitrary but same colour — fix with gadgets!



not i and i' both green

Sketch of Reduction

- Given FPHP instance for bipartite graph of left degree k
- Order available holes $\mathcal{N}(i) = \{j_{i,1}, \ldots, j_{i,c}\}$ for every pigeon i
- Vertex i coloured with colour c $\Leftrightarrow$ pigeon i flies to hole $j_{i,c}$
- $j_{i,c} = j_{i',c'}$ $\Rightarrow$ can't colour i by c and i' by c' simultaneously
- *Almost* colouring, except forbidding specific colour pair (c, c') instead of arbitrary but same colour — fix with gadgets!

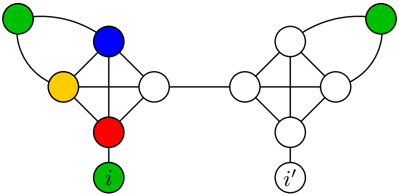


not i and i' both green

Colouring i green...

Sketch of Reduction

- Given FPHP instance for bipartite graph of left degree k
- Order available holes $\mathcal{N}(i) = \{j_{i,1}, \ldots, j_{i,c}\}$ for every pigeon i
- Vertex i coloured with colour c $\Leftrightarrow$ pigeon i flies to hole $j_{i,c}$
- $j_{i,c} = j_{i',c'}$ $\Rightarrow$ can't colour i by c and i' by c' simultaneously
- *Almost* colouring, except forbidding specific colour pair (c, c') instead of arbitrary but same colour — fix with gadgets!

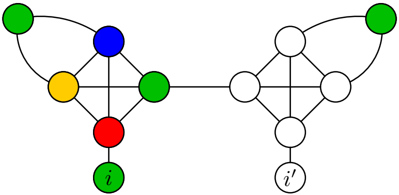


not i and i' both green

Colouring i green forces left 4-clique use all other colours

Sketch of Reduction

- Given FPHP instance for bipartite graph of left degree k
- Order available holes $\mathcal{N}(i) = \{j_{i,1}, \ldots, j_{i,c}\}$ for every pigeon i
- Vertex i coloured with colour c $\Leftrightarrow$ pigeon i flies to hole $j_{i,c}$
- $j_{i,c} = j_{i',c'}$ $\Rightarrow$ can't colour i by c and i' by c' simultaneously
- *Almost* colouring, except forbidding specific colour pair (c, c') instead of arbitrary but same colour — fix with gadgets!

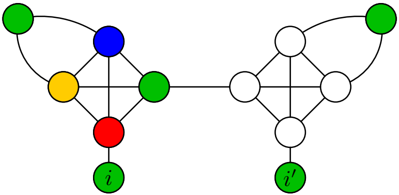


not i and i' both green

Colouring i green forces left 4-clique use all other colours making rightmost node green

Sketch of Reduction

- Given FPHP instance for bipartite graph of left degree k
- Order available holes $\mathcal{N}(i) = \{j_{i,1}, \ldots, j_{i,c}\}$ for every pigeon i
- Vertex i coloured with colour c $\Leftrightarrow$ pigeon i flies to hole $j_{i,c}$
- $j_{i,c} = j_{i',c'}$ $\Rightarrow$ can't colour i by c and i' by c' simultaneously
- *Almost* colouring, except forbidding specific colour pair (c, c') instead of arbitrary but same colour — fix with gadgets!



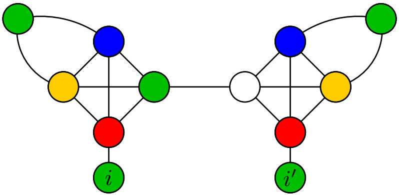
not i and i' both green

Colouring i green forces left 4-clique use all other colours making rightmost node green

Symmetric argument in right subgadget...

Sketch of Reduction

- Given FPHP instance for bipartite graph of left degree k
- Order available holes $\mathcal{N}(i) = \{j_{i,1}, \ldots, j_{i,c}\}$ for every pigeon i
- Vertex i coloured with colour c $\Leftrightarrow$ pigeon i flies to hole $j_{i,c}$
- $j_{i,c} = j_{i',c'}$ $\Rightarrow$ can't colour i by c and i' by c' simultaneously
- *Almost* colouring, except forbidding specific colour pair (c, c') instead of arbitrary but same colour — fix with gadgets!



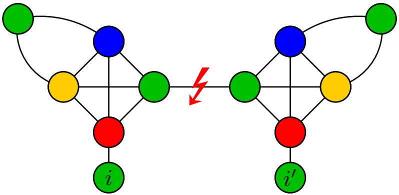
not i and i' both green

Colouring i green forces left 4-clique use all other colours making rightmost node green

Symmetric argument in right subgadget...

Sketch of Reduction

- Given FPHP instance for bipartite graph of left degree k
- Order available holes $\mathcal{N}(i) = \{j_{i,1}, \ldots, j_{i,c}\}$ for every pigeon i
- Vertex i coloured with colour c $\Leftrightarrow$ pigeon i flies to hole $j_{i,c}$
- $j_{i,c} = j_{i',c'}$ $\Rightarrow$ can't colour i by c and i' by c' simultaneously
- *Almost* colouring, except forbidding specific colour pair (c, c') instead of arbitrary but same colour — fix with gadgets!



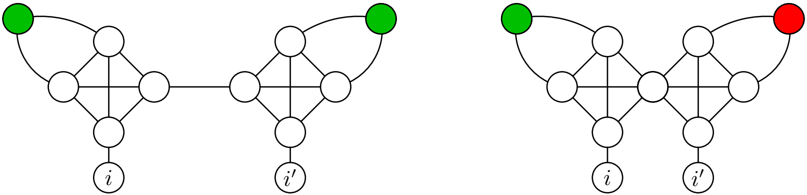
not i and i' both green

Colouring i green forces left 4-clique use all other colours making rightmost node green

Symmetric argument in right subgadget $\Rightarrow$ **contradiction**

Sketch of Reduction

- Given FPHP instance for bipartite graph of left degree k
- Order available holes $\mathcal{N}(i) = \{j_{i,1}, \ldots, j_{i,c}\}$ for every pigeon i
- Vertex i coloured with colour c $\Leftrightarrow$ pigeon i flies to hole $j_{i,c}$
- $j_{i,c} = j_{i',c'}$ $\Rightarrow$ can't colour i by c and i' by c' simultaneously
- *Almost* colouring, except forbidding specific colour pair (c, c') instead of arbitrary but same colour — fix with gadgets!



not i and i' both green

not i green and i' red

Open Problems

- Prove PC degree lower bounds for other formulas

Open Problems

- Prove PC degree lower bounds for other formulas
  - independent set formulas
  - average-case for graph colouring formulas
  - dense linear ordering formulas

Open Problems

- Prove PC degree lower bounds for other formulas
  - independent set formulas
  - average-case for graph colouring formulas
  - dense linear ordering formulas
- Prove size lower bounds via technique that doesn't use degree

Open Problems

- Prove PC degree lower bounds for other formulas
  - independent set formulas
  - average-case for graph colouring formulas
  - dense linear ordering formulas
- Prove size lower bounds via technique that doesn't use degree
  - k-clique formulas
  - weak pigeonhole principle formulas ($\geq n^2$ pigeons)

Open Problems

- Prove PC degree lower bounds for other formulas
  - independent set formulas
  - average-case for graph colouring formulas
  - dense linear ordering formulas
- Prove size lower bounds via technique that doesn't use degree
  - k-clique formulas
  - weak pigeonhole principle formulas ($\geq n^2$ pigeons)
- Find truly general framework capturing all PC degree bounds

Open Problems

- Prove PC degree lower bounds for other formulas
  - independent set formulas
  - average-case for graph colouring formulas
  - dense linear ordering formulas
- Prove size lower bounds via technique that doesn't use degree
  - k-clique formulas
  - weak pigeonhole principle formulas ($\geq n^2$ pigeons)
- Find truly general framework capturing all PC degree bounds
  - We generalize only part of [AR03]
  - Cannot handle characteristic-dependent bounds à la [BGIP01]
  - Combination of [AR03] and [MN15] might give lower bounds for even colouring formulas [Mar06, VEG$^+$18]

Take-away Message

Generalized method for width and degree lower bounds

- Unified framework for most previous lower bounds
- Highlights similarities and differences between resolution and polynomial calculus
- Exponential polynomial calculus size lower bound for
  - functional PHP
  - graph colouring

Future directions

- Extend techniques further to other tricky formulas
- Develop non-degree-based size lower bound techniques

Take-away Message

Generalized method for width and degree lower bounds

- Unified framework for most previous lower bounds
- Highlights similarities and differences between resolution and polynomial calculus
- Exponential polynomial calculus size lower bound for
  - functional PHP
  - graph colouring

Future directions

- Extend techniques further to other tricky formulas
- Develop non-degree-based size lower bound techniques

Thank you for your attention!

References I

[ABRW02] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM Journal on Computing*, 31(4):1184–1211, 2002. Preliminary version in *STOC '00*.

[ALN16] Albert Atserias, Massimo Lauria, and Jakob Nordström. Narrow proofs may be maximally long. *ACM Transactions on Computational Logic*, 17(3):19:1–19:30, May 2016. Preliminary version in *CCC '14*.

[AR03] Michael Alekhnovich and Alexander A. Razborov. Lower bounds for polynomial calculus: Non-binomial case. *Proceedings of the Steklov Institute of Mathematics*, 242:18–35, 2003. Available at `http://people.cs.uchicago.edu/~razborov/files/misha.pdf`. Preliminary version in *FOCS '01*.

[BCMM05] Paul Beame, Joseph C. Culberson, David G. Mitchell, and Cristopher Moore. The resolution complexity of random graph k-colorability. *Discrete Applied Mathematics*, 153(1-3):25–47, December 2005.

[BG01] María Luisa Bonet and Nicola Galesi. Optimality of size-width tradeoffs for resolution. *Computational Complexity*, 10(4):261–276, December 2001. Preliminary version in *FOCS '99*.

References II

[BGIP01] Samuel R. Buss, Dima Grigoriev, Russell Impagliazzo, and Toniann Pitassi. Linear gaps between degrees for the polynomial calculus modulo distinct primes. *Journal of Computer and System Sciences*, 62(2):267–289, March 2001. Preliminary version in *CCC '99*.

[BI99] Eli Ben-Sasson and Russell Impagliazzo. Random CNF's are hard for the polynomial calculus. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS '99)*, pages 415–421, October 1999.

[BKPS02] Paul Beame, Richard Karp, Toniann Pitassi, and Michael Saks. The efficiency of resolution and Davis-Putnam procedures. *SIAM Journal on Computing*, 31(4):1048–1075, 2002. Preliminary versions of these results appeared in *FOCS '96* and *STOC '98*.

[BW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, March 2001. Preliminary version in *STOC '99*.

References III

[CEI96] Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC '96)*, pages 174–183, May 1996.

[CS88] Vašek Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, October 1988.

[DLMM08] Jesús A. De Loera, Jon Lee, Peter N. Malkin, and Susan Margulies. Hilbert's Nullstellensatz and an algorithm for proving combinatorial infeasibility. In *Proceedings of the 21st International Symposium on Symbolic and Algebraic Computation (ISSAC '08)*, pages 197–206, July 2008.

[DLMM11] Jesús A. De Loera, Jon Lee, Peter N. Malkin, and Susan Margulies. Computing infeasibility certificates for combinatorial problems through Hilbert's Nullstellensatz. *Journal of Symbolic Computation*, 46(11):1260–1283, November 2011.

References IV

[DLMO09] Jesús A. De Loera, Jon Lee, Susan Margulies, and Shmuel Onn. Expressing combinatorial problems by systems of polynomial equations and Hilbert's Nullstellensatz. *Combinatorics, Probability and Computing*, 18(04):551–582, July 2009.

[DMP+15] Jesús A. De Loera, Susan Margulies, Michael Pernpeintner, Eric Riedl, David Rolnick, Gwen Spencer, Despina Stasi, and Jon Swenson. Graph-coloring ideals: Nullstellensatz certificates, Gröbner bases for chordal graphs, and hardness of Gröbner bases. In *Proceedings of the 40th International Symposium on Symbolic and Algebraic Computation (ISSAC '15)*, pages 133–140, July 2015.

[GL10] Nicola Galesi and Massimo Lauria. Optimality of size-degree trade-offs for polynomial calculus. *ACM Transactions on Computational Logic*, 12(1):4:1–4:22, November 2010.

[Hak85] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39(2-3):297–308, August 1985.

References V

[IPS99] Russell Impagliazzo, Pavel Pudlák, and Jiří Sgall. Lower bounds for the polynomial calculus and the Gröbner basis algorithm. *Computational Complexity*, 8(2):127–144, 1999.

[LLO16] Bo Li, Benjamin Lowenstein, and Mohamed Omar. Low degree Nullstellensatz certificates for 3-colorability. *The Electronic Journal of Combinatorics*, 23(1), January 2016.

[LN17] Massimo Lauria and Jakob Nordström. Graph colouring is hard for algorithms based on Hilbert's Nullstellensatz and Gröbner bases. In *Proceedings of the 32nd Annual Computational Complexity Conference (CCC '17)*, volume 79 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:20, July 2017.

[Mar06] Klas Markström. Locality and hard SAT-instances. *Journal on Satisfiability, Boolean Modeling and Computation*, 2(1-4):221–227, 2006.

[MN14] Mladen Mikša and Jakob Nordström. Long proofs of (seemingly) simple formulas. In *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT '14)*, volume 8561 of *Lecture Notes in Computer Science*, pages 121–137. Springer, July 2014.

References VI

[MN15] Mladen Mikša and Jakob Nordström. A generalized method for proving polynomial calculus degree lower bounds. In *Proceedings of the 30th Annual Computational Complexity Conference (CCC '15)*, volume 33 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 467–487, June 2015.

[Raz02] Alexander A. Razborov. Proof complexity of pigeonhole principles. In *5th International Conference on Developments in Language Theory, (DLT '01), Revised Papers*, volume 2295 of *Lecture Notes in Computer Science*, pages 100–116. Springer, July 2002.

[Raz14] Alexander A. Razborov. Possible research directions. List of open problems (in proof complexity and other areas) available at `http://people.cs.uchicago.edu/~razborov/teaching/`, 2014.

[Rii93] Søren Riis. *Independence in Bounded Arithmetic*. PhD thesis, University of Oxford, 1993.

[Urq87] Alasdair Urquhart. Hard examples for resolution. *Journal of the ACM*, 34(1):209–219, January 1987.

References VII

[VEG+18] Marc Vinyals, Jan Elffers, Jesús Giráldez-Cru, Stephan Gocht, and Jakob Nordström. In between resolution and cutting planes: A study of proof systems for pseudo-Boolean SAT solving. In *Proceedings of the 21st International Conference on Theory and Applications of Satisfiability Testing (SAT '18)*, volume 10929 of *Lecture Notes in Computer Science*, pages 292–310. Springer, July 2018.