Propositional proof complexity
Mini-tutorial

Edward A. Hirsch

http://logic.pdmi.ras.ru/~hirsch

Steklov Institute of Mathematics at St.Petersburg

Estonian Theory Days — October 3, 2009

Propositional proof complexity
Mini-tutorial

Edward A. Hirsch

http://logic.pdmi.ras.ru/~hirsch

Steklov Institute of Mathematics at St.Petersburg

Estonian Theory Days — October 3, 2009

- Proof systems — definitions and examples.
- A lower bound.
- Connection to optimal algorithms.
- Connection to disjoint NP pairs.

Proof systems

Definition (Cook, Reckhow, 70s)

A proof system for language L is a polynomial-time surjective mapping $\Pi\colon \{0,1\}^* \to L$.

Proof systems

Definition (Cook, Reckhow, 70s)

A proof system for language L is a polynomial-time surjective mapping $\Pi \colon \{0,1\}^* \to L$.

We consider proof systems for the language of Boolean tautologies **TAUT** (propositional proof systems).

Definition (almost equivalent)

A propositional proof system is a polynomial-time verification procedure V such that

$$F \text{ is a tautology} \iff \exists \pi \ V(F, \pi) = \text{``}\mathtt{OK}\text{''}.$$

Proof systems

Definition (Cook, Reckhow, 70s)

A proof system for language L is a polynomial-time surjective mapping $\Pi\colon \{0,1\}^* \to L$.

We consider proof systems for the language of Boolean tautologies **TAUT** (propositional proof systems).

Definition (almost equivalent)

A propositional proof system is a polynomial-time verification procedure V such that

$$F \text{ is a tautology} \iff \exists \pi\ V(F,\pi) = \text{``}\mathtt{OK}\text{''}.$$

Every algorithm for **TAUT** yields a proof system, but not vice versa.

Proof systems

Definition (Cook, Reckhow, 70s)

A proof system for language L is a polynomial-time surjective mapping $\Pi\colon \{0,1\}^* \to L$.

We consider proof systems for the language of Boolean tautologies **TAUT** (propositional proof systems).

Definition (almost equivalent)

A propositional proof system is a polynomial-time verification procedure V such that

$$F \text{ is a tautology} \iff \exists \pi\ V(F,\pi) = \text{``}\mathtt{OK}\text{''}.$$

Every algorithm for **TAUT** yields a proof system, but not vice versa.

Fact

NP $=$ **co**-**NP** *iff there is a proof system that has a polynomial-size proof for every tautology.*

Example: Resolution

- Consider the negation of input formula F; it has no satisfying assignments iff F is a tautology.
- W.l.o.g. it is in CNF, e.g.,

$$(a \vee b \vee \neg c) \wedge (a \vee c) \wedge (a \vee \neg b) \wedge (\neg a).$$

- Resolution is the inference of logical consequences:

$$\frac{(x \vee \alpha) \qquad (\neg x \vee \beta)}{\alpha \vee \beta.}$$

- We finish when we infer the empty disjunction (i.e., contradiction).
- Any such inference is a valid resolution proof (can be very long!).

Example: Nullstellensatz

- Boolean variable $\mapsto$ 0/1 variable.
- $\neg x \mapsto (1 - x)$.
- clause $a \vee b \vee c \vee \ldots \mapsto$ polynomial $(1 - a)(1 - b)(1 - c) \ldots$.
- Add polynomials $x^2 - x$ for every variable x.
- Boolean formula is unsatisfiable iff
  all these polynomials p_i have no common roots.
- Hilbert's Nullstellensatz: hence, there are polynomials g_i such that $\sum_i p_i g_i = 1$ (constant polynomial).
- This set $\{g_i\}_i$ is a proof!

Example: Cutting Plane

- Boolean variable $\mapsto$ 0/1 variable.
- $\neg x \mapsto (1 - x)$.
- clause $a \vee b \vee c \vee \ldots \mapsto$ inequality $a + b + c \geq 1$.
- Add trivial inequalities $x \geq 0$ and $1 \geq x$.
- Boolean formula is unsatisfiable iff
  the system of inequalities has integer solutions.
- Infer logical consequences:

$$\frac{A \geq 0 \qquad B \geq 0}{kA + \ell B \geq 0}; \qquad \frac{kA \geq \ell}{A \geq \lceil \ell / k \rceil}$$

  for positive integers k, ℓ.
- We finish when we infer $-1 \geq 0$ (i.e., contradiction).

Pigeon-hole principle

- Variable x_{ij} — i-th pigeon is in j-th hole ($1 \leq i \leq n+1, 1 \leq j \leq n$).
- $\bigvee_j x_{ij}$
  — i-th pigeon is sitting somewhere,
- $\neg x_{ij} \vee \neg x_{i'j}$
  — two pigeons cannot sit together.

Pigeon-hole principle

- Variable x_{ij} — i-th pigeon is in j-th hole ($1 \le i \le n+1, 1 \le j \le n$).
- $\bigvee_j x_{ij}$
  — i-th pigeon is sitting somewhere,
- $\neg x_{ij} \vee \neg x_{i'j}$
  — two pigeons cannot sit together.
- No polynomial-size Resolution proofs [Haken, 80s].

Pigeon-hole principle

- Variable x_{ij} — i-th pigeon is in j-th hole $(1 \leq i \leq n+1, 1 \leq j \leq n)$.
- $\bigvee_j x_{ij}$
  — i-th pigeon is sitting somewhere,
- $\neg x_{ij} \vee \neg x_{i'j}$
  — two pigeons cannot sit together.
- No polynomial-size Resolution proofs [Haken, 80s].
- Polynomial-size Cutting Plane proof:

Pigeon-hole principle

- Variable x_{ij} — i-th pigeon is in j-th hole $(1 \leq i \leq n+1, 1 \leq j \leq n)$.
- $\sum_j x_{ij} \geq 1 \qquad (*)$
  — i-th pigeon is sitting somewhere,
- $x_{ij} + x_{i'j} \leq 1$
  — two pigeons cannot sit together.
- No polynomial-size Resolution proofs [Haken, 80s].
- Polynomial-size Cutting Plane proof:

Pigeon-hole principle

- Variable x_{ij} — i-th pigeon is in j-th hole ($1 \leq i \leq n+1, 1 \leq j \leq n$).
- $\sum_j x_{ij} \geq 1 \qquad (*)$
  — i-th pigeon is sitting somewhere,
- $x_{ij} + x_{i'j} \leq 1$
  — two pigeons cannot sit together.
- No polynomial-size Resolution proofs [Haken, 80s].
- Polynomial-size Cutting Plane proof:

$$\cfrac{\cfrac{\cfrac{\cfrac{x_{ij} + x_{i'j} \leq 1 \qquad x_{i'j} + x_{i''j} \leq 1 \qquad x_{i''j} + x_{ij} \leq 1}{2(x_{ij} + x_{i'j} + x_{i''j}) \leq 3}}{x_{ij} + x_{i'j} + x_{i''j} \leq 1}}{\ldots}}{\sum_i x_{ij} \leq 1}$$

Pigeon-hole principle

- Variable x_{ij} — i-th pigeon is in j-th hole $(1 \leq i \leq n+1, 1 \leq j \leq n)$.
- $\sum_j x_{ij} \geq 1 \qquad (*)$
  — i-th pigeon is sitting somewhere,
- $x_{ij} + x_{i'j} \leq 1$
  — two pigeons cannot sit together.
- No polynomial-size Resolution proofs [Haken, 80s].
- Polynomial-size Cutting Plane proof:

$$\cfrac{\cfrac{\cfrac{x_{ij} + x_{i'j} \leq 1 \qquad x_{i'j} + x_{i''j} \leq 1 \qquad x_{i''j} + x_{ij} \leq 1}{2(x_{ij} + x_{i'j} + x_{i''j}) \leq 3}}{x_{ij} + x_{i'j} + x_{i''j} \leq 1}}{\cfrac{\dots}{\sum_i x_{ij} \leq 1}}$$

- In total $\sum_{ij} x_{ij} \leq m$.

Pigeon-hole principle

- Variable x_{ij} — i-th pigeon is in j-th hole ($1 \leq i \leq n+1, 1 \leq j \leq n$).
- $\sum_j x_{ij} \geq 1 \qquad (*)$
  — i-th pigeon is sitting somewhere,
- $x_{ij} + x_{i'j} \leq 1$
  — two pigeons cannot sit together.
- No polynomial-size Resolution proofs [Haken, 80s].
- Polynomial-size Cutting Plane proof:

$$\dfrac{\dfrac{\dfrac{x_{ij} + x_{i'j} \leq 1 \qquad x_{i'j} + x_{i''j} \leq 1 \qquad x_{i''j} + x_{ij} \leq 1}{2(x_{ij} + x_{i'j} + x_{i''j}) \leq 3}}{\dfrac{x_{ij} + x_{i'j} + x_{i''j} \leq 1}{\dots}}}{\sum_i x_{ij} \leq 1}$$

- In total $\sum_{ij} x_{ij} \leq m$.
- But $(*)$ gives $\sum_{ji} x_{ij} \geq m+1$.

Simulation and Optimal system

Definition

A proof system S simulates a proof system W (written $S \leq W$) iff S-proofs are at most as long as W-proofs (up to a polynomial p):

$$\forall F \in \textbf{TAUT} \ |\text{shortest } S\text{-proof of } F| \leq p(|\text{shortest } W\text{-proof of } F|).$$

Simulation and Optimal system

Definition

A proof system S simulates a proof system W (written $S \leq W$) iff S-proofs are at most as long as W-proofs (up to a polynomial p):

$$\forall F \in \textbf{TAUT} \ |\text{shortest } S\text{-proof of } F| \leq p(|\text{shortest } W\text{-proof of } F|).$$

S strictly simulates W (written $S < W$) if in addition $W \not\leq S$.
For example, Cutting Plane strictly simulates Resolution.

Simulation and Optimal system

Definition

A proof system S simulates a proof system W (written $S \leq W$) iff S-proofs are at most as long as W-proofs (up to a polynomial p):

$$\forall F \in \textbf{TAUT} \; |\text{shortest } S\text{-proof of } F| \leq p(|\text{shortest } W\text{-proof of } F|).$$

S strictly simulates W (written $S < W$) if in addition $W \not\leq S$.
For example, Cutting Plane strictly simulates Resolution.

Definition

p-simulation $\leq_p$ is a constructive version:
For any s-size W-proof one can compute a $p(s)$-size S-proof in polynomial time.

Simulation and Optimal system

Definition

A proof system S simulates a proof system W (written $S \leq W$) iff S-proofs are at most as long as W-proofs (up to a polynomial p):

$$\forall F \in \mathbf{TAUT}\ |\text{shortest } S\text{-proof of } F| \leq p(|\text{shortest } W\text{-proof of } F|).$$

S strictly simulates W (written $S < W$) if in addition $W \not\leq S$.
For example, Cutting Plane strictly simulates Resolution.

Definition

p-simulation $\leq_p$ is a constructive version:
For any s-size W-proof one can compute a $p(s)$-size S-proof in polynomial time.

Definition

(p-)optimal proof system is the smallest element in this lattice.

Simulation and Optimal system

Definition

A proof system S simulates a proof system W (written $S \leq W$) iff S-proofs are at most as long as W-proofs (up to a polynomial p):

$$\forall F \in \textbf{TAUT} \ |\text{shortest } S\text{-proof of } F| \leq p(|\text{shortest } W\text{-proof of } F|).$$

S strictly simulates W (written $S < W$) if in addition $W \not\leq S$.
For example, Cutting Plane strictly simulates Resolution.

Definition

p-simulation $\leq_p$ is a constructive version:
For any s-size W-proof one can compute a $p(s)$-size S-proof in polynomial time.

Definition

(p-)optimal proof system is the smallest element in this lattice.

Does it exist?..

A lower bound for Resolution

Clique is a monotone function: if a graph does not have a clique, its subgraphs don't. Thus it is computable by monotone circuits (no negations).

Theorem (Razborov, 80s; Pudlak, 90s)

Polynomial-size monotone Boolean (and even real) circuits cannot compute Clique. They cannot even distinguish m-cliques from complete $(m-1)$*-partite graphs, where* $m = \lfloor (n/\log n)^{2/3}/8 \rfloor$*,* n *is the number of vertices.*

Our strategy: short proof $\mapsto$ small monotone Boolean circuit.

Clique-coloring formula

Claims that there is an m-clique in an $(m-1)$-colorable graph with n vertices.
Variables:

- q_{ki} maps number k to vertex i,
- e_{ij} stays for the edge $\{i, j\}$,
- $c_{i\ell}$ colors vertex i by color ℓ.

Clauses:

- $\bigvee_{i=1}^{n} q_{ki}$
  — there is a mapping of $\{1, \ldots, m\}$ to the graph,
- $\neg q_{ki} \vee \neg q_{k'i}$
  — it is injective,
- $\neg q_{ki} \vee \neg q_{k',j} \vee e_{ij}$
  — its image is indeed a clique,
- $\bigvee_{\ell=1}^{m-1} c_{i\ell}$
  — each vertex is colored,
- $\neg e_{ij} \vee \neg c_{i\ell} \vee \neg c_{j\ell}$.
  — the coloring is correct.

Monotone interpolation [Pudlák, 90s]

- For every fixed graph $\{e_{ij}\}_{i,j}$,
  we have only $q_{\ldots}$-clauses (clique) and $c_{\ldots}$-clauses (coloring).
- Either there is no clique or there is no coloring.
  Deciding between the two alternatives distinguishes m-cliques from $(m-1)$-colorable graphs.
- **The main thing to prove:** A short proof of the initial formula gives a small monotone circuit for this problem, which does not exist by Razborov's theorem.

Optimal algorithms

Definition

A is an optimal algorithm for language L if for any other algorithm A' there is a polynomial p such that $\forall x \in L$

$$\mathrm{time}_A(x) \le p(\mathrm{time}_{A'}(x) + |x|).$$

Levin's optimal algorithm for **SAT**:
run "in parallel" all possible algorithms outputting satisfying assignments; check the results and output as soon as a correct one found.

Remark

Levin's algorithm is not for **TAUT**.

Optimal algorithms vs Optimal proof systems

Theorem (Krajíček, Pudlák, 89)

$\exists$ *p-optimal proof system iff* $\exists$ *an optimal algorithm for* ***TAUT***.

Optimal algorithms vs Optimal proof systems

Theorem (Krajíček, Pudlák, 89)

$\exists$ *p-optimal proof system iff* $\exists$ *an optimal algorithm for* ***TAUT****.*

$\Longleftarrow$:

- Optimal algorithm is polynomial-time on every polynomial-time recognizable set of tautologies.

Optimal algorithms vs Optimal proof systems

Theorem (Krajíček, Pudlák, 89)

*$\exists$ p-optimal proof system iff $\exists$ an optimal algorithm for **TAUT**.*

$\Longleftarrow$:

- Optimal algorithm is polynomial-time on every polynomial-time recognizable set of tautologies.
- For every proof system Π, one can write in polynomial time the tautology $\mathrm{Con}_{\Pi,n}$ meaning the system is correct for formulas of size n.

Optimal algorithms vs Optimal proof systems

Theorem (Krajíček, Pudlák, 89)

$\exists$ *p-optimal proof system iff* $\exists$ *an optimal algorithm for* ***TAUT***.

$\Longleftarrow$:

- Optimal algorithm is polynomial-time on every polynomial-time recognizable set of tautologies.
- For every proof system Π, one can write in polynomial time the tautology $\mathrm{Con}_{\Pi,n}$ meaning the system is correct for formulas of size n.
- Thus optimal algorithm is polynomial-time on $\mathrm{Con}_{\Pi,n}$.

Optimal algorithms vs Optimal proof systems

Theorem (Krajíček, Pudlák, 89)

$\exists$ *p-optimal proof system iff* $\exists$ *an optimal algorithm for* ***TAUT***.

$\Longleftarrow$:

- Optimal algorithm is polynomial-time on every polynomial-time recognizable set of tautologies.
- For every proof system Π, one can write in polynomial time the tautology $\mathrm{Con}_{\Pi,n}$ meaning the system is correct for formulas of size n.
- Thus optimal algorithm is polynomial-time on $\mathrm{Con}_{\Pi,n}$.
- Now an optimal proof of F of size n includes
  - Description of proof system Π;
  - Description of the execution of the optimal algorithm on $\mathrm{Con}_{\Pi,n}$;
  - A Π-proof of F.

Optimal algorithms vs Optimal proof systems

Theorem (Krajíček, Pudlák, 89)

$\exists$ *p-optimal proof system iff* $\exists$ *an optimal algorithm for* ***TAUT****.*

$\Longrightarrow$:

- Let Π be a p-optimal proof system.

Optimal algorithms vs Optimal proof systems

Theorem (Krajíček, Pudlák, 89)

$\exists$ *p-optimal proof system iff* $\exists$ *an optimal algorithm for* ***TAUT***.

$\Longrightarrow$:

- Let Π be a p-optimal proof system.
- Optimal algorithm runs in parallel
  all algorithms A_i trying to produce a Π-proof of F.
- The "proof" is checked by Π. Say "yes" if it's valid.

Optimal algorithms vs Optimal proof systems

Theorem (Krajíček, Pudlák, 89)

$\exists$ p-optimal proof system iff $\exists$ an optimal algorithm for ***TAUT****.*

$\Longrightarrow$:

- Let Π be a p-optimal proof system.
- Optimal algorithm runs in parallel
  all algorithms A_i trying to produce a Π-proof of F.
- The "proof" is checked by Π. Say "yes" if it's valid.
- Since Π is p-optimal, for every algorithm A there is a polynomial-time transformation f of its execution into a Π-proof. Thus A together with f are listed in $\{A_i\}_i$.

Heuristic optimal algorithm for **TAUT**

- Allow randomized algorithms (with bounded error).
- Allow small number[1] of false theorems (unbounded error there).
- Then an optimal algorithm does exist:
  - Run all possible algorithms "in parallel".
  - First check each algorithm by generating random non-theorems and making sure the algorithm does not lie quickly.
  - Say "yes" as soon as the first good algorithm says so.
- Unfortunately, the equivalence with optimal proof systems is unknown to work.

[1] According to a samplable distribution on non-theorems.

Disjoint NP pairs

- Just a pair (A, B) of two disjoint sets $A, B \in \mathbf{NP}$.
- The problem is to separate A from B: given x, decide between the two alternatives $x \in A$ vs $x \in B$ (if it is outside both, say anything).
- Reduction $(A, B) \to (C, D)$:
  polynomial-time f such that $f(A) \subseteq C$, $f(B) \subseteq D$.
- Are there complete ones? Unknown.

Where they come from

Example

Consider a bitwise cryptosystem.
$A = \{\text{possible codes of } 0\}$,
$B = \{\text{possible codes of } 1\}$.
One hopes it's impossible to separate in polynomial time!

Example

Consider a proof system Π for $\mathbf{TAUT}$.
$\overline{\mathbf{TAUT}}_* = \{(F, 1^t) \mid F \in \overline{\mathbf{TAUT}}\}$,
$\mathbf{REF}_\Pi = \{(F, 1^t) \mid F \in \mathbf{TAUT},\ \text{there is a } \Pi\text{-proof of } F \text{ of size} \leq t\}$.
Separation gives automatization!

Simulation vs Reduction

Theorem

Simulation $S \leq W$ yields reduction of the **NP** *pair* $(\overline{\textbf{\textit{TAUT}}}_*, \textbf{REF}_W) \to (\overline{\textbf{\textit{TAUT}}}_*, \textbf{REF}_S)$.

Simulation vs Reduction

Theorem

Simulation $S \leq W$ *yields reduction of the* **NP** *pair* $(\overline{\boldsymbol{TAUT}}_*, \mathbf{REF}_W) \rightarrow (\overline{\boldsymbol{TAUT}}_*, \mathbf{REF}_S)$.

Optimal proof system yields complete **NP** *pair.*

Simulation vs Reduction

Theorem

Simulation $S \leq W$ yields reduction of the **NP** *pair* $(\overline{\boldsymbol{TAUT}}_*, \mathbf{REF}_W) \to (\overline{\boldsymbol{TAUT}}_*, \mathbf{REF}_S)$.

Optimal proof system yields complete **NP** *pair.*

- Consider $(F, 1^t) \in \mathbf{REF}_W$.
- One needs to transform $(F, 1^t)$ claiming t-size Π_1-proof into $(F, 1^s)$ claiming s-size Π_2-proof.

Simulation vs Reduction

Theorem

Simulation $S \leq W$ yields reduction of the **NP** *pair* $(\overline{\boldsymbol{TAUT}}_*, \mathbf{REF}_W) \to (\overline{\boldsymbol{TAUT}}_*, \mathbf{REF}_S)$.

Optimal proof system yields complete **NP** *pair.*

- Consider $(F, 1^t) \in \mathbf{REF}_W$.
- One needs to transform $(F, 1^t)$ claiming t-size Π_1-proof into $(F, 1^s)$ claiming s-size Π_2-proof.
- We know that s polynomially depends on t.
  Just plug in this polynomial p: $(F, 1^t) \to (F, 1^{p(t)})$.

Simulation vs Reduction

Theorem

Simulation $S \leq W$ yields reduction of the **NP** *pair $(\overline{\boldsymbol{TAUT}}_*, \mathbf{REF}_W) \to (\overline{\boldsymbol{TAUT}}_*, \mathbf{REF}_S)$.*

Optimal proof system yields complete **NP** *pair.*

- Consider $(F, 1^t) \in \mathbf{REF}_W$.
- One needs to transform $(F, 1^t)$ claiming t-size Π_1-proof into $(F, 1^s)$ claiming s-size Π_2-proof.
- We know that s polynomially depends on t.
  Just plug in this polynomial p: $(F, 1^t) \to (F, 1^{p(t)})$.
- For $(F, 1^t) \in \overline{\mathbf{TAUT}}_*$, the change in $1^{\cdots}$ does not mater.

Open questions

1. Lower bounds for proof systems.
   - Frege-style systems (work with formulas), Gentzen system.
   - Semialgebraic systems (quadratic inequalities; disjunctions of linear inequalities).
2. Upper bounds for proof systems.
   - We can solve $\mathbf{3} - \mathbf{SAT}$ in time $O(1.3^n)$;
     what's about proof size — it could be better?
3. Optimal proof system.
   - Show a collapse if there is one.
   - Construct a heuristic optimal proof system.
     - Vice versa, show that equivalence to heuristic optimal algorithms will not work.