Application of Communication Complexity: Time/Memory trade-offs in SAT-solving

Theoretical Computer Science Group
KTH Royal Institute of Technology

DD2441 Autumn 2012:
Communication Complexity
October 8, 2012

The SAT Problem

CNF $\quad \bigwedge_{i=1}^{c} \bigvee_{j=1}^{d_i} \ell_{i,j} \qquad \ell_{i,j} = \begin{cases} x \\ \neg x \end{cases}$ for some variable x

SAT $\quad (\neg x \vee \neg y \vee z) \wedge (x \vee z) \wedge (\neg x \vee y)$

e.g. $x = \bot, y = \top, z = \top$

UNSAT $\quad (\neg x \vee y) \wedge \neg y \wedge (x \vee \neg z) \wedge (y \vee z)$

The SAT Problem in Theory and Practice

- SAT NP-complete and so probably intractable in worst case
- But enormous progress on applied algorithms last 10-15 years
- Surprising fact: State-of-the-art SAT solvers can deal with real-world instances containing millions of variables

SAT solvers

SAT satisfying assignment

UNSAT proof of unsatisfiability

Key computational resources for modern solvers are:

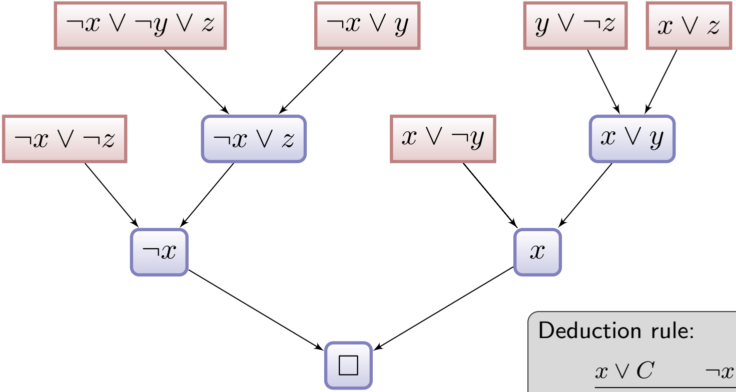
- Running Time
- Memory

Propositional proof systems

Deterministic polynomial time $P(\cdot, \cdot)$

- if $F \in \textsc{Unsat}$ then $P(F, \pi) = 1$ for some $\pi \in \{0, 1\}^*$
- if $F \notin \textsc{Unsat}$ then $P(F, \pi) = 0$ for all $\pi \in \{0, 1\}^*$

Propositional proof systems: example



Deduction rule:

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

SAT solver $\longrightarrow$ Proof system

SAT-Solver(F)=`unsat`	$\longrightarrow$	refutation of F

Resolution based

DPLL	$\longrightarrow$	tree-like resolution
Clause Learning	$\longrightarrow$	fragments of (regular) resolution
CL + Restarts	$\longrightarrow$	resolution

Algebraic

CryptoMiniSat	$\longrightarrow$	fragments of PCR on $GF(2)$
PolyBoRi	$\longrightarrow$	PC on $GF(2)$

Geometric

Gomory Cuts	$\longrightarrow$	cutting planes
Cplex	$\longrightarrow$	cutting planes+others

SAT solver $\longrightarrow$ Proof system

- Constructive (almost deterministic) algorithms
- Key resources for solvers: time and memory
- Ideally minimize simultaneously

SAT solver $\longrightarrow$ Proof system

- Constructive (almost deterministic) algorithms
- Key resources for solvers: time and memory
- Ideally minimize simultaneously

- Study proofs, i.e., nondeterministic algorithms
- Complexity measures: proof size and proof space
- Lower bounds for optimal algorithms

SAT solver $\longrightarrow$ Proof system

- Constructive (almost deterministic) algorithms
- Key resources for solvers: time and memory
- Ideally minimize simultaneously

- Study proofs, i.e., nondeterministic algorithms
- Complexity measures: proof size and proof space
- Lower bounds for optimal algorithms

Hope to understand potential and limitation of SAT solvers by studying corresponding proof systems

In this and next lecture

Objective

We want to analyse the relation between memory and time in SAT solving.

- can we solve SAT quickly?
- can we solve SAT with little memory?
- can we do both simultaneously?

Main Idea

Go to proof complexity: a **short proof** in **small space** gives an efficient protocol for an hard communication problem.

Based on research paper:

The content of this and next lecture is based on the paper

> *On the Virtue of Succinct Proofs: Amplifying Communication Complexity Hardness to Time-Space Trade-offs in Proof Complexity.*
>
> — by Trinh Huynh and Jakob Nordström (STOC '12)

Outline

1. Intro to proof Complexity
   - Computational model
   - Proof Systems
2. Main Theorem
3. Proof Ingredients
   - Search problem for UNSAT
   - Lifting
   - Critical Block Sensitivity
   - Pebbling
4. Conclusion

1. Intro to Proof Complexity

Computational model for small space proofs

Some Terminology and Notation

- 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 $F = C_1 \wedge \cdots \wedge C_m$: conjunction of clauses
- k-CNF formula: all clauses of size $\leq k = \mathcal{O}(1)$
- Goal: Refute given CNF formula (i.e., prove it is unsatisfiable)
- Refer to clauses of CNF formula as axioms
  (as opposed to conclusions derived from these clauses)
- All formulas in this talk are k-CNFs
  (cleanest and most interesting case)

The Theoretical Model

- Proof system operates with lines of some syntactic form
- Proof/refutation is "presented on blackboard"
- Derivation steps:
  - Write down axiom clauses of CNF formula being refuted (as encoded by proof system)
  - Infer new lines by deductive rules of proof system
  - Erase lines not currently needed (to save space on blackboard)
- Refutation ends when contradiction is derived

The Theoretical Model

- Proof system operates with lines of some syntactic form
- Proof/refutation is "presented on blackboard"
- Derivation steps:
  - Write down axiom clauses of CNF formula being refuted (as encoded by proof system)
  - Infer new lines by deductive rules of proof system
  - Erase lines not currently needed (to save space on blackboard)
- Refutation ends when contradiction is derived

The Theoretical Model

- Proof system operates with lines of some syntactic form
- Proof/refutation is “presented on blackboard”
- Derivation steps:
  - Write down axiom clauses of CNF formula being refuted (as encoded by proof system)
  - Infer new lines by deductive rules of proof system
  - Erase lines not currently needed (to save space on blackboard)
- Refutation ends when contradiction is derived

$$\boxed{x \lor \overline{y} \lor z}$$

The Theoretical Model

- Proof system operates with lines of some syntactic form
- Proof/refutation is "presented on blackboard"
- Derivation steps:
  - Write down axiom clauses of CNF formula being refuted (as encoded by proof system)
  - Infer new lines by deductive rules of proof system
  - Erase lines not currently needed (to save space on blackboard)
- Refutation ends when contradiction is derived

$$\boxed{\begin{array}{l} x \vee \overline{y} \vee z \\ \overline{z} \vee \overline{u} \vee w \\ \\ \end{array}}$$

The Theoretical Model

- Proof system operates with lines of some syntactic form
- Proof/refutation is “presented on blackboard”
- Derivation steps:
  - Write down axiom clauses of CNF formula being refuted (as encoded by proof system)
  - Infer new lines by deductive rules of proof system
  - Erase lines not currently needed (to save space on blackboard)
- Refutation ends when contradiction is derived

$$x \vee \overline{y} \vee z$$
$$\overline{z} \vee \overline{u} \vee w$$
$$x \vee \overline{y} \vee \overline{u} \vee w$$

The Theoretical Model

- Proof system operates with lines of some syntactic form
- Proof/refutation is "presented on blackboard"
- Derivation steps:
  - Write down axiom clauses of CNF formula being refuted (as encoded by proof system)
  - Infer new lines by deductive rules of proof system
  - Erase lines not currently needed (to save space on blackboard)
- Refutation ends when contradiction is derived

$$x \vee \overline{y} \vee z$$
$$\overline{z} \vee \overline{u} \vee w$$
$$x \vee \overline{y} \vee \overline{u} \vee w$$

The Theoretical Model

- Proof system operates with lines of some syntactic form
- Proof/refutation is "presented on blackboard"
- Derivation steps:
  - Write down axiom clauses of CNF formula being refuted (as encoded by proof system)
  - Infer new lines by deductive rules of proof system
  - Erase lines not currently needed (to save space on blackboard)
- Refutation ends when contradiction is derived

$$\boxed{\begin{array}{l} \overline{z} \lor \overline{u} \lor w \\ x \lor \overline{y} \lor \overline{u} \lor w \end{array}}$$

The Theoretical Model

- Proof system operates with lines of some syntactic form
- Proof/refutation is "presented on blackboard"
- Derivation steps:
  - Write down axiom clauses of CNF formula being refuted (as encoded by proof system)
  - Infer new lines by deductive rules of proof system
  - Erase lines not currently needed (to save space on blackboard)
- Refutation ends when contradiction is derived

$$\overline{z} \vee \overline{u} \vee w$$
$$x \vee \overline{y} \vee \overline{u} \vee w$$

The Theoretical Model

- Proof system operates with lines of some syntactic form
- Proof/refutation is "presented on blackboard"
- Derivation steps:
  - Write down axiom clauses of CNF formula being refuted (as encoded by proof system)
  - Infer new lines by deductive rules of proof system
  - Erase lines not currently needed (to save space on blackboard)
- Refutation ends when contradiction is derived

$$\boxed{x \lor \overline{y} \lor \overline{u} \lor w}$$

Complexity Measures: Length, Size and Space

Length

$\#$ derivation steps

Size

$\approx$ total $\#$ symbols in proof counted with repetitions

Space

$\approx$ max size of blackboard to carry out proof
(e.g., space 3 for this blackboard)

$$\boxed{\begin{array}{l} x \vee \overline{y} \vee z \\ \overline{z} \vee \overline{u} \vee w \\ x \vee \overline{y} \vee \overline{u} \vee w \end{array}}$$

Proof Systems

Resolution

Basis for the most successful SAT solvers to date
(DPLL method plus clause learning; a.k.a. CDCL)

Lines in refutation are disjunctive clauses

Resolution rule $\dfrac{C \vee x \qquad D \vee \overline{x}}{C \vee D}$

Polynomial Calculus (or PCR)

Clauses interpreted as polynomial equations over finite field
E.g., $x \vee y \vee \overline{z}$ translated to $x'y'z = 0$
Show no common root by deriving $1 = 0$

Boolean axioms $\dfrac{}{x^2 - x = 0}$

Negation $\dfrac{}{x + x' = 1}$

Linear combination $\dfrac{p = 0 \qquad q = 0}{\alpha p + \beta q = 0}$

Multiplication $\dfrac{p = 0}{xp = 0}$

Cutting Planes

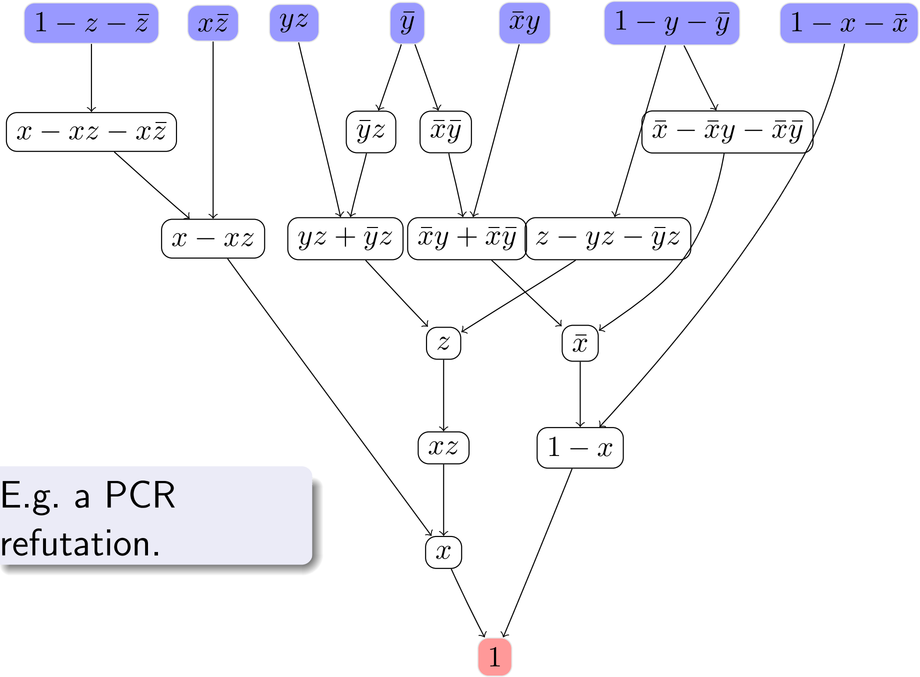
Clauses interpreted as linear inequalities
E.g., $x \vee y \vee \overline{z}$ translated to $x + y + (1 - z) \geq 1$
Show inconsistent by deriving $0 \geq 1$

Variable axioms $$\frac{}{0 \leq x \leq 1}$$

Multiplication $$\frac{\sum a_i x_i \geq A}{\sum c a_i x_i \geq cA}$$

Addition $$\frac{\sum a_i x_i \geq A \quad \sum b_i x_i \geq B}{\sum (a_i + b_i) x_i \geq A + B}$$

Division $$\frac{\sum c a_i x_i \geq A}{\sum a_i x_i \geq \lceil A/c \rceil}$$



E.g. a PCR refutation.

E.g. a PCR proof in the blackboard model

$$\emptyset \cdots \to \begin{bmatrix} xz + yz \\ xz - 1 \\ \end{bmatrix}$$

E.g. a PCR proof in the blackboard model

$$\emptyset \cdots \rightarrow \begin{bmatrix} xz + yz \\ xz - 1 \\ \end{bmatrix} \rightarrow \begin{bmatrix} xz + yz \\ xz - 1 \\ 1 + yz \end{bmatrix}$$

E.g. a PCR proof in the blackboard model

$$\emptyset \cdots \to \begin{bmatrix} xz + yz \\ xz - 1 \\ \end{bmatrix} \to \begin{bmatrix} xz + yz \\ xz - 1 \\ 1 + yz \end{bmatrix} \to \begin{bmatrix} xz + yz \\ \text{—} \\ 1 + yz \end{bmatrix}$$

E.g. a PCR proof in the blackboard model

$$\emptyset \cdots \rightarrow \begin{bmatrix} xz + yz \\ xz - 1 \\ \end{bmatrix} \rightarrow \begin{bmatrix} xz + yz \\ xz - 1 \\ 1 + yz \end{bmatrix} \rightarrow \begin{bmatrix} xz + yz \\ \\ 1 + yz \end{bmatrix} \rightarrow \begin{bmatrix} xz + yz \\ \color{red}{x^2 - x} \\ 1 + yz \end{bmatrix} \cdots$$

E.g. a PCR proof in the blackboard model

$$\emptyset \cdots \to \begin{bmatrix} xz + yz \\ xz - 1 \\ \end{bmatrix} \to \begin{bmatrix} xz + yz \\ xz - 1 \\ 1 + yz \end{bmatrix} \to \begin{bmatrix} xz + yz \\ \\ 1 + yz \end{bmatrix} \to \begin{bmatrix} xz + yz \\ x^2 - x \\ 1 + yz \end{bmatrix} \cdots \to [1]$$

Length, Size and Space

Length
is the number of lines in the proof, or equivalently the number of vertices in the directed acyclic graph representing the proof.

Size
$\approx$ sum over all proof lines of the number of symbols in a line (counted with repetitions). In PCR a line can have a super-polynomial number of symbols, while it is not the case in Resolution or CP.

Space

- Resolution: max # of clauses in a blackboard during a proof.
- Cutting Planes: max # of inequalities in a blackboard during a proof.
- PCR: max # of monomials in a blackboard during a proof.

Known results

- Resolution

Length	several lower bounds	[U '87; CS '88]
Space	optimal lower bounds	[T '99; ABRW '00; BG '03]
Trade-offs	strong size-space trade-offs	[BN '11; BBI '12]

- Cutting Planes

Length	one lower bound	[P '97]
Space	nothing is known	
Trade-offs	very limited trade-offs	[**HN '12**]

- Polynomial Calculus

Length	exponential lower bounds on size	[AR '01]
Space	recent progress	[ABRW '00; FLNRT '12; BG '13]
Trade-offs	limited size-space trade-offs	[**HN '12**; BNT '12]

2. Main Theorem

Main Theorem

Theorem (Huynh, Nordström (STOC '12))

There are k-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that

- *resolution can refute F_n in length $\mathcal{O}(n)$ (and hence so can polynomial calculus and cutting planes)*
- *any polynomial calculus or cutting planes refutation of F_n in length L and space s must have*

$$s \log L \gtrsim \sqrt[4]{n}$$

3. Proof Ingredients

Proof Ingredients

- A communication problem based on UNSAT CNFs.
- Lifting of search problem (to make it harder)
- Critical block sensitivity (source of hardness)
- Pebbling formulas (large block sensitivity)

Search problem on UNSAT CNF

Given

- $F = \wedge_i C_i$ an unsatisfiable CNF;
- α an assignment;

find:

- C_i such that α falsifies C_i

Search problem on UNSAT CNF

Given

- $F = \wedge_i C_i$ an unsatisfiable CNF;
- α an assignment;

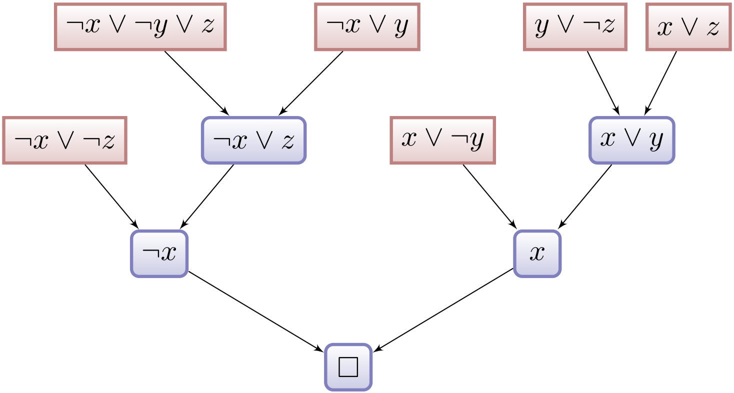
find:

- C_i such that α falsifies C_i

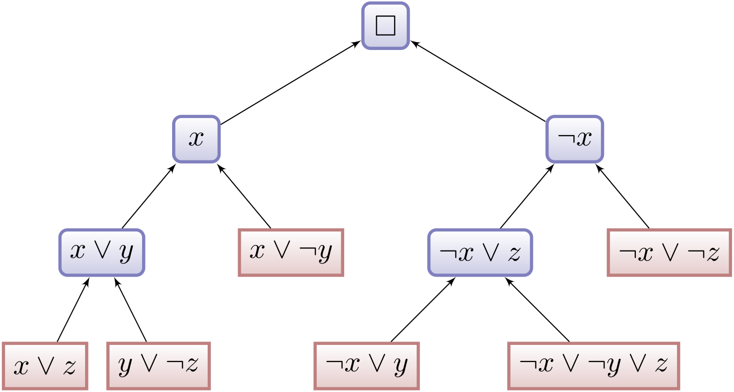
Basic intuition

An "efficient refutation" for F gives an "efficient method" to solve the search problem on F.

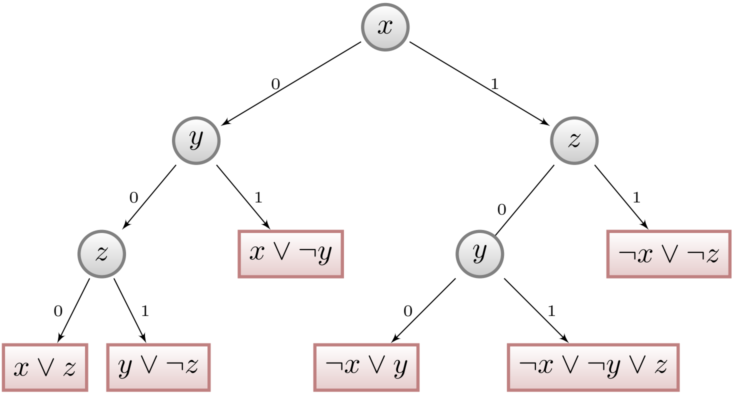
From proofs to search procedures (example)



From proofs to search procedures (example)



From proofs to search procedures (example)



Search problem as a Communication problem

- Alice gets part of the assignment α and her own private randomness
- Bob gets the other part of α, and his own private randomness

Falsified clause search problem $Search(F)$

Input: Assignment α to $Vars(F)$ split between Alice and Bob

Output: Clause $C \in F$ such that $\alpha(C) = 0$

Search problem as a Communication problem

- Alice gets part of the assignment α and her own private randomness
- Bob gets the other part of α, and his own private randomness

Falsified clause search problem $Search(F)$

Input: Assignment α to $Vars(F)$ split between Alice and Bob

Output: Clause $C \in F$ such that $\alpha(C) = 0$

CC point of view

How many bit do they need to exchange to solve the search Problem?

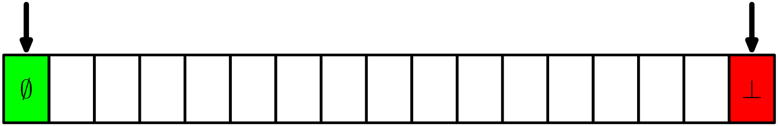
Succinct Refutations Yield Efficient Protocols

Evaluate blackboard configurations of a refutation of F under α

$\emptyset$																$\perp$

Succinct Refutations Yield Efficient Protocols

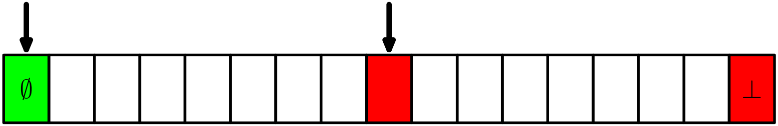
Evaluate blackboard configurations of a refutation of F under α



Use binary search to find transition from true to false blackboard

Succinct Refutations Yield Efficient Protocols

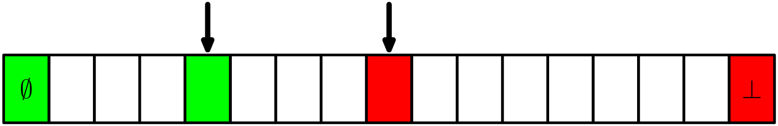
Evaluate blackboard configurations of a refutation of F under α



Use binary search to find transition from true to false blackboard

Succinct Refutations Yield Efficient Protocols

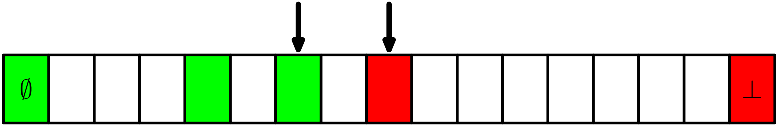
Evaluate blackboard configurations of a refutation of F under α



Use binary search to find transition from true to false blackboard

Succinct Refutations Yield Efficient Protocols

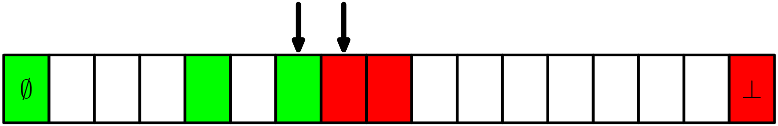
Evaluate blackboard configurations of a refutation of F under α



Use binary search to find transition from true to false blackboard

Succinct Refutations Yield Efficient Protocols

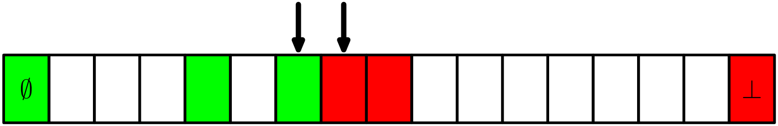
Evaluate blackboard configurations of a refutation of F under α



Use binary search to find transition from true to false blackboard

Succinct Refutations Yield Efficient Protocols

Evaluate blackboard configurations of a refutation of F under α

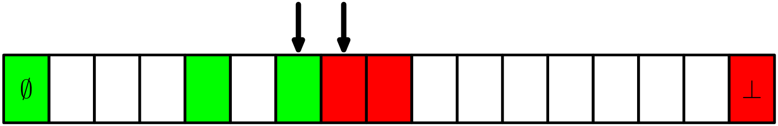


Use binary search to find transition from true to false blackboard

Must happen when $C \in F$ written down — answer to $Search(F)$

Succinct Refutations Yield Efficient Protocols

Evaluate blackboard configurations of a refutation of F under α



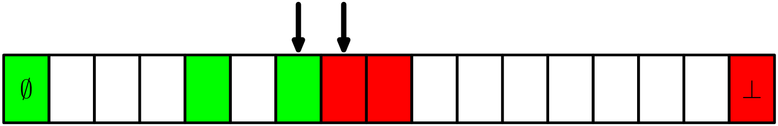
Use binary search to find transition from true to false blackboard

Must happen when $C \in F$ written down — answer to $Search(F)$

Refutation length $L \Rightarrow$ evaluate $\log L$ blackboards

Succinct Refutations Yield Efficient Protocols

Evaluate blackboard configurations of a refutation of F under α



Use binary search to find transition from true to false blackboard

Must happen when $C \in F$ written down — answer to $Search(F)$

Refutation length $L \Rightarrow$ evaluate $\log L$ blackboards

Refutation space $s \Rightarrow$ max $\approx s$ bits of communication per blackboard

How to evaluate blackboards (Polynomial Calculus)

For each monomial Alice and Bob evaluate their part and send the values to each other.

Observation

A polynomial calculus refutation of length L and monomial space s implies a deterministic communication protocol for $Search(F)$ of cost

$$\mathcal{O}\big(s \log L\big)$$

How to evaluate blackboards (Cutting Planes I)

Alice has variables x_i and Bob has variables y_j, they evaluate the line

$$\sum_i a_i x_i + \sum_j b_j y_j \leq c$$

- Alice computes $A = \sum_i a_i x_i$;
- Bob computes $B = \sum_j c - b_j y_j$;
- they compute $GT(A, B)$ paying $\mathcal{O}(\log^2 n \log(s \log L))$ bits.

How to evaluate blackboards (Cutting Planes II)

Observation

A cutting planes refutation of length L and space s implies a randomized communication protocol for $Search(F)$ of cost

$$\mathcal{O}\big(s \log L \log(s \log L) \log^2 n\big).$$

Proof.

1. Each inequality evaluation costs $\mathcal{O}(\log^2 n \log(s \log L))$ bits.
2. Each evaluation fails with probability at most (say)

$$\frac{1}{4s \log L}$$

3. At most $s \log L$ inequalities are evaluated.

□

Q: how do we get hard search problem?

Q: how do we get hard search problem?

A: we “lift” moderately hard search problem.

Formal definition of a search problem

It is a set

$$S \subseteq \{0,1\}^m \times A$$

such that for any $q \in \{0,1\}^m$ there exists $(q, a) \in S$.

- q is a "query"
- a is an "answer" (one of possibly many)

Lifting of a search problem

Start with search problem
$S \in \{0,1\}^m \times A$

Lifting of a search problem

Start with search problem
$S \in \{0,1\}^m \times A$

We want a query q for S

$q := \boxed{}\boxed{}\boxed{}$

Lifting of a search problem

Start with search problem
$S \in \{0,1\}^m \times A$

We want a query q for S

Construct the query using inputs
$x \in \{0,1\}^{\ell m}$ and $y \in [\ell]^m$

y_1	y_2	y_3

$x_{1,1}$	$x_{1,2}$	$x_{2,1}$	$x_{2,2}$	$x_{3,1}$	$x_{3,2}$

$q :=$

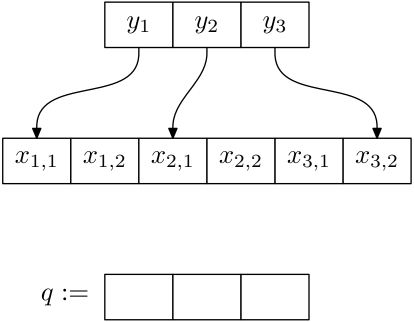
Lifting of a search problem

Start with search problem
$S \in \{0,1\}^m \times A$

We want a query q for S

Construct the query using inputs
$x \in \{0,1\}^{\ell m}$ and $y \in [\ell]^m$

Alice's y-variables determine. . .



Lifting of a search problem

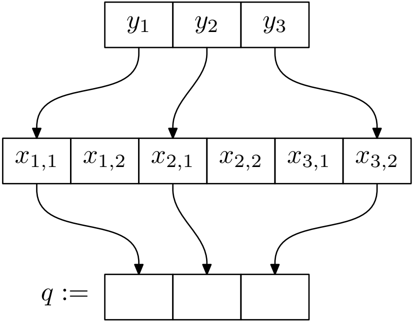
Start with search problem
$S \in \{0,1\}^m \times A$

We want a query q for S

Construct the query using inputs
$x \in \{0,1\}^{\ell m}$ and $y \in [\ell]^m$

Alice's y-variables determine...

...which of Bob's x-bits is in q



Lifting of a search problem

Start with search problem
$S \in \{0,1\}^m \times A$

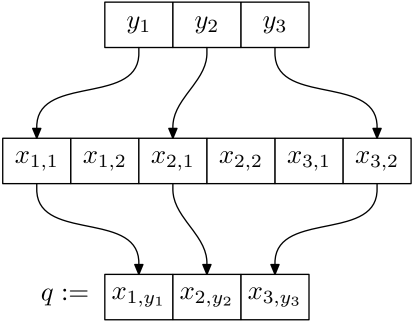
We want a query q for S

Construct the query using inputs
$x \in \{0,1\}^{\ell m}$ and $y \in [\ell]^m$

Alice's y-variables determine...

...which of Bob's x-bits is in q

Length-ℓ lifting of S defined as



$$(x, y, q) \in Lift_\ell(S) \iff (\langle x_{1,y_1}, \ldots, x_{m,y_m} \rangle, q) \in S$$

Lifting does not create hardness

Lifting per se does not make a search problem hard.
It allows to better exploit the complexity of the original search problem.

Source of hardness: block sensitivity

Block Sensitivity for functions

Block sensitivity of f on α: $\#$ disjoint blocks of α that flip f if flipped

$$f\left(\boxed{0}\boxed{1}\boxed{1}\boxed{0}\boxed{1}\boxed{0}\boxed{1}\boxed{0}\boxed{0}\boxed{0}\boxed{1}\boxed{0}\right) = 0$$

Block Sensitivity for functions

Block sensitivity of f on α: $\#$ disjoint blocks of α that flip f if flipped

$$f\left(\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}\hline 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ \hline\end{array}\right) = 1$$

Block Sensitivity for functions

Block sensitivity of f on α: # disjoint blocks of α that flip f if flipped

$$f\left(\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}\hline 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ \hline\end{array}\right) = 0$$

Block Sensitivity for functions

Block sensitivity of f on α: $\#$ disjoint blocks of α that flip f if flipped

$$f\left(\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}\hline 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline\end{array}\right) = 1$$

Block Sensitivity for functions

Block sensitivity of f on α: $\#$ disjoint blocks of α that flip f if flipped

$$f\left(\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}\hline 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ \hline\end{array}\right) = 0$$

Block Sensitivity for functions

Block sensitivity of f on α: $\#$ disjoint blocks of α that flip f if flipped

$$f\left(\,0\ \boxed{1\ 1\ 0}\ \boxed{1\ 0\ 1}\ 0\ \boxed{1}\ 0\ 1\ 0\,\right) = 1$$

Block Sensitivity for functions

Block sensitivity of f on α: $\#$ disjoint blocks of α that flip f if flipped

$$f\left(0\ \boxed{1\ 1\ 0}\ \boxed{1\ 0\ 1}\ 0\ \boxed{0}\ 0\ 1\ 0\right) = 0$$

Block Sensitivity for functions

Block sensitivity of f on α: $\#$ disjoint blocks of α that flip f if flipped

$$f\left(0\ \boxed{1\ 1\ 0}\ \boxed{1\ 0\ 1}\ 0\ \boxed{0}\ 0\ 1\ 0\right) = 0$$

$bs(f, \alpha) = 3$ in this example

Block Sensitivity for functions

Block sensitivity of f on α: # disjoint blocks of α that flip f if flipped

$$f\left(0\ \boxed{1\ 1\ 0}\ \boxed{1\ 0\ 1}\ 0\ \boxed{0}\ 0\ 1\ 0\right) = 0$$

$bs(f,\alpha) = 3$ in this example

$bs(f,\mathcal{A})$: largest block sensitivity for any α in subset $\mathcal{A}$ of inputs

Block Sensitivity for search problems

f solves search problem $S \subseteq \{0,1\}^m \times Q$ if it holds that $(\alpha, f(\alpha)) \in S$

Block Sensitivity for search problems

f solves search problem $S \subseteq \{0,1\}^m \times Q$ if it holds that $(\alpha, f(\alpha)) \in S$

We focus on critical assignment of S (i.e. only one answer).

Block Sensitivity for search problems

f solves search problem $S \subseteq \{0,1\}^m \times Q$ if it holds that $(\alpha, f(\alpha)) \in S$

We focus on critical assignment of S (i.e. only one answer).

$bs_{crit}(S)$: block sensitivity over critical assignments $\mathcal{A}$ of best f solving S

Block Sensitivity for search problems

f solves search problem $S \subseteq \{0,1\}^m \times Q$ if it holds that $(\alpha, f(\alpha)) \in S$

We focus on critical assignment of S (i.e. only one answer).

$bs_{crit}(S)$: block sensitivity over critical assignments $\mathcal{A}$ of best f solving S

- on $\mathcal{A}$ the problem S is like a function;
- the assignments obtained by flipping blocks, may not be critical.

Lifting and Critical Block Sensitivity (I)

Lemma 1 (informal)

If critical block sensitivity of search problem S is large, then communication complexity of lifted search problem $Lift(S)$ is large

Lifting and Critical Block Sensitivity (II)

Lemma 1 (more formal version)

Suppose $S \subseteq \{0,1\}^m \times Q$ is a search problem and $\ell \geq 3$. Then any consistent randomized protocol solving $Lift_\ell(S)$, where Alice receives the selector y-variables and Bob receives the main x-variables, requires $\Omega(bs_{crit}(S))$ bits of communication.

Proof is by

- information theory tools
- direct sum theorem à la [BJKS04]

Consistent protocol?

A two-player randomized protocol Π for S is such that

$$\Pr[(x, y, \Pi(x, y)) \in S] > \frac{2}{3}.$$

Consistent protocol?

A two-player randomized protocol Π for S is such that

$$\Pr[(x, y, \Pi(x, y)) \in S] > \frac{2}{3}.$$

A consistent protocol instead:

$$\exists (x, y, q) \in S \text{ such that } \Pr[(x, y) = q] > \frac{2}{3}.$$

It does matter since consistent protocols are weaker in some cases.

Putting the Pieces Together

- Encode lifting of search problem for CNF as new formula $\mathit{Lift}(F)$ (as in [Beame, Huynh & Pitassi '10])

Putting the Pieces Together

- Encode lifting of search problem for CNF as new formula $Lift(F)$ (as in [Beame, Huynh & Pitassi '10])
- Small length **AND** small space refutation of $Lift(F)$
  $\Rightarrow$ efficient communication protocol for $Search(Lift(F))$

Putting the Pieces Together

- Encode lifting of search problem for CNF as new formula $\mathit{Lift}(F)$ (as in [Beame, Huynh & Pitassi '10])
- Small length **AND** small space refutation of $\mathit{Lift}(F)$
  $\Rightarrow$ efficient communication protocol for $\mathit{Search}(\mathit{Lift}(F))$
- Protocol for $\mathit{Search}(\mathit{Lift}(F))$
  $\Rightarrow$ use to solve $\mathit{Lift}(\mathit{Search}(F))$ — easy

Putting the Pieces Together

- Encode lifting of search problem for CNF as new formula $\mathit{Lift}(F)$ (as in [Beame, Huynh & Pitassi '10])
- Small length **AND** small space refutation of $\mathit{Lift}(F)$
  $\Rightarrow$ efficient communication protocol for $\mathit{Search}(\mathit{Lift}(F))$
- Protocol for $\mathit{Search}(\mathit{Lift}(F))$
  $\Rightarrow$ use to solve $\mathit{Lift}(\mathit{Search}(F))$ — easy
- But communication complexity of lifted search problem lower-bounded by critical block sensitivity (Lemma 1)

Putting the Pieces Together

- Encode lifting of search problem for CNF as new formula $\mathit{Lift}(F)$ (as in [Beame, Huynh & Pitassi '10])
- Small length **AND** small space refutation of $\mathit{Lift}(F)$
  $\Rightarrow$ efficient communication protocol for $\mathit{Search}(\mathit{Lift}(F))$
- Protocol for $\mathit{Search}(\mathit{Lift}(F))$
  $\Rightarrow$ use to solve $\mathit{Lift}(\mathit{Search}(F))$ — easy
- But communication complexity of lifted search problem lower-bounded by critical block sensitivity (Lemma 1)

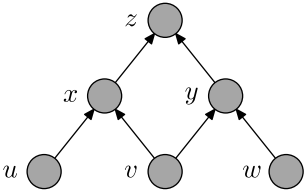
We need a CNF F such that $\mathit{Search}(F)$ has large block sensitivity

Pebbling contradictions on pyramids

Pebbling Contradiction

CNF formulas encoding pebble game played on DAG G

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

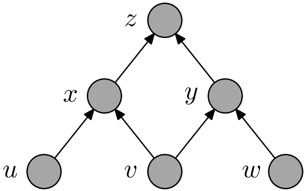


- sources are true
- truth propagates upwards
- but sink is false

Pebbling Contradiction

CNF formulas encoding pebble game played on DAG G

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

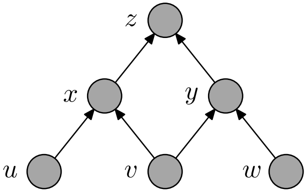


- sources are true
- truth propagates upwards
- but sink is false

Pebbling Contradiction

CNF formulas encoding pebble game played on DAG G

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

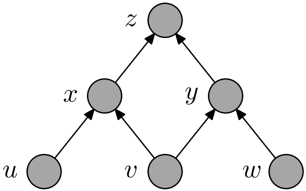


- sources are true
- truth propagates upwards
- but sink is false

Pebbling Contradiction

CNF formulas encoding pebble game played on DAG G

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

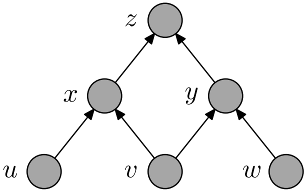


- sources are true
- truth propagates upwards
- but sink is false

Pebbling Contradiction

CNF formulas encoding pebble game played on DAG G

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$



- sources are true
- truth propagates upwards
- but sink is false

Appeared in various contexts in [Bonet et al. '98, Raz & McKenzie '99, Ben-Sasson & Wigderson '99] and other papers

Used to study size and space in resolution in [N. '06, N. & Håstad '08, Ben-Sasson & N. '08, '11]

Pebbling and Time-Space Relation

Questions about time-space trade-offs fundamental in theoretical computer science

Pebbling and Time-Space Relation

Questions about time-space trade-offs fundamental in theoretical computer science

In particular, well-studied (and well-understood) for pebble games modelling calculations described by DAGs ([Cook & Sethi '76] and many others)

- Time needed for calculation: # pebbling moves
- Space needed for calculation: max # pebbles required

Pebbling and Time-Space Relation

Questions about time-space trade-offs fundamental in theoretical computer science

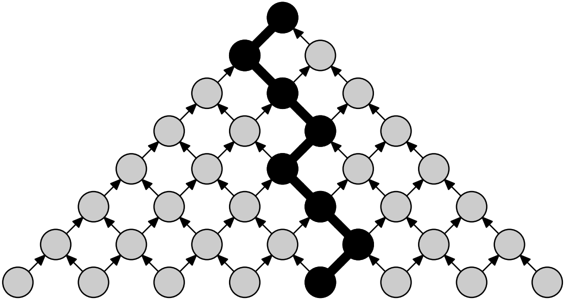
In particular, well-studied (and well-understood) for pebble games modelling calculations described by DAGs ([Cook & Sethi '76] and many others)

- Time needed for calculation: # pebbling moves
- Space needed for calculation: max # pebbles required

Pebbling game and Resolution

- pebbling time $\approx$ refutation length
- pebbling space $\approx$ clause space
- time/space trade-offs $\approx$ proof length/space trade-offs

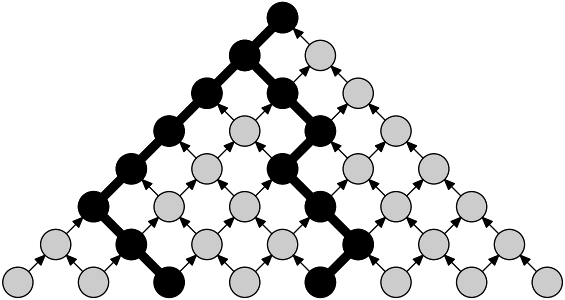
Critical Assignments for Pyramid Pebbling Contradiction



Focus on critical assignment setting:

- vertices on one source-to-sink path P false
- all other vertices true (so $\mathrm{source}(P)$ only correct answer)

Critical Assignments for Pyramid Pebbling Contradiction



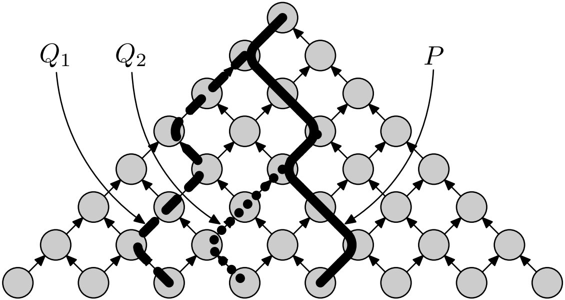
Focus on critical assignment setting:

- vertices on one source-to-sink path P false
- all other vertices true (so $\mathrm{source}(P)$ only correct answer)

Bicritical assignments falsify two different paths

$\Rightarrow$ two possible correct answers

Path Graph



Build graph G such that

- vertices = source-to-sink paths P
- edge (P, Q) only if P and Q merge and stay together
- in addition, if (P, Q_1) and (P, Q_2) edges, then $Q_1 \cap Q_2 \subseteq P$
- G is undirected — (P, Q) edge only if (Q, P) edge

Dense Path Graph $\Rightarrow$ High Critical Block Sensitivity

Lemma 2

If $\exists$ path graph G with average degree d, then falsified clause search problem for pebbling formula has critical block sensitivity $\geq d/2$

Dense Path Graph ⇒ High Critical Block Sensitivity

Lemma 2

If $\exists$ path graph G with average degree d, then falsified clause search problem for pebbling formula has critical block sensitivity $\geq d/2$

Proof:

- Orient G based on function f solving search problem

Dense Path Graph $\Rightarrow$ High Critical Block Sensitivity

Lemma 2

If $\exists$ path graph G with average degree d, then falsified clause search problem for pebbling formula has critical block sensitivity $\geq d/2$

Proof:

- Orient G based on function f solving search problem
- If f answers $\mathrm{source}(Q)$ for bicritical (P, Q), direct edge $P \rightarrow Q$

Dense Path Graph $\Rightarrow$ High Critical Block Sensitivity

Lemma 2

If $\exists$ path graph G with average degree d, then falsified clause search problem for pebbling formula has critical block sensitivity $\geq d/2$

Proof:

- Orient G based on function f solving search problem
- If f answers $\mathrm{source}(Q)$ for bicritical (P, Q), direct edge $P \rightarrow Q$
- Some P must have outdegree $\geq d/2$

Dense Path Graph ⇒ High Critical Block Sensitivity

Lemma 2

If $\exists$ path graph G with average degree d, then falsified clause search problem for pebbling formula has critical block sensitivity $\geq d/2$

Proof:

- Orient G based on function f solving search problem
- If f answers $\mathrm{source}(Q)$ for bicritical (P, Q), direct edge $P \to Q$
- Some P must have outdegree $\geq d/2$
- When P flipped to bicritical (P, Q_i) for $P \to Q_i$, then f changes

Dense Path Graph ⇒ High Critical Block Sensitivity

Lemma 2

If $\exists$ path graph G with average degree d, then falsified clause search problem for pebbling formula has critical block sensitivity $\geq d/2$

Proof:

- Orient G based on function f solving search problem
- If f answers $\mathrm{source}(Q)$ for bicritical (P, Q), direct edge $P \to Q$
- Some P must have outdegree $\geq d/2$
- When P flipped to bicritical (P, Q_i) for $P \to Q_i$, then f changes
- Hence critical block sensitivity $\geq d/2$, Q.E.D.

Dense Path Graph $\Rightarrow$ High Critical Block Sensitivity

Lemma 2

If $\exists$ path graph G with average degree d, then falsified clause search problem for pebbling formula has critical block sensitivity $\geq d/2$

Proof:

- Orient G based on function f solving search problem
- If f answers $\mathrm{source}(Q)$ for bicritical (P, Q), direct edge $P \rightarrow Q$
- Some P must have outdegree $\geq d/2$
- When P flipped to bicritical (P, Q_i) for $P \rightarrow Q_i$, then f changes
- Hence critical block sensitivity $\geq d/2$, Q.E.D.

Lemma 3

For pyramid on n vertices, can get average degree $\Omega\left(\sqrt[4]{n}\right)$

CNFs with large critical block sensitivity search problem

Corollary 4

Search problems for pebbling formulas of pyramid graphs have critical block sensitivity $\Omega\left(\sqrt[4]{n}\right)$.

4. Conclusion

Theorem (Huynh, Nordström (STOC '12))

There are k-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that

- *resolution can refute F_n in length $\mathcal{O}(n)$ (and hence so can polynomial calculus and cutting planes)*
- *any polynomial calculus or cutting planes refutation of F_n in length L and space s must have*

$$s \log L \gtrsim \sqrt[4]{n}$$

Proof.

□

Theorem (Huynh, Nordström (STOC '12))

There are k-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that

- *resolution can refute F_n in length $\mathcal{O}(n)$ (and hence so can polynomial calculus and cutting planes)*
- *any polynomial calculus or cutting planes refutation of F_n in length L and space s must have*

$$s \log L \gtrapprox \sqrt[4]{n}$$

Proof.

1. Pick P_n to be pebbling formula over the pyramind of n vertices

□

Theorem (Huynh, Nordström (STOC '12))

There are k-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that

- *resolution can refute F_n in length $\mathcal{O}(n)$ (and hence so can polynomial calculus and cutting planes)*
- *any polynomial calculus or cutting planes refutation of F_n in length L and space s must have*

$$s \log L \gtrsim \sqrt[4]{n}$$

Proof.

1. Pick P_n to be pebbling formula over the pyramind of n vertices
2. Set $F_n := \mathit{Lift}_3(P_n)$. F_n has resolution refutation of length $\mathcal{O}(n)$

□

Theorem (Huynh, Nordström (STOC '12))

There are k-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that

- *resolution can refute F_n in length $\mathcal{O}(n)$ (and hence so can polynomial calculus and cutting planes)*
- *any polynomial calculus or cutting planes refutation of F_n in length L and space s must have*

$$s \log L \gtrsim \sqrt[4]{n}$$

Proof.

1. Pick P_n to be pebbling formula over the pyramind of n vertices
2. Set $F_n := \mathit{Lift}_3(P_n)$. F_n has resolution refutation of length $\mathcal{O}(n)$
3. In CP and PC it holds that $s \log L \gtrsim bs_{crit}(\mathit{Search}(P_n))$ (Lemma 1)

$\square$

Theorem (Huynh, Nordström (STOC '12))

There are k-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that

- *resolution can refute F_n in length $\mathcal{O}(n)$ (and hence so can polynomial calculus and cutting planes)*
- *any polynomial calculus or cutting planes refutation of F_n in length L and space s must have*

$$s \log L \gtrapprox \sqrt[4]{n}$$

Proof.

1. Pick P_n to be pebbling formula over the pyramind of n vertices
2. Set $F_n := \mathit{Lift}_3(P_n)$. F_n has resolution refutation of length $\mathcal{O}(n)$
3. In CP and PC it holds that $s \log L \gtrapprox bs_{crit}(\mathit{Search}(P_n))$ (Lemma 1)
4. $bs_{crit}(\mathit{Search}(P_n)) \geq \sqrt[4]{n}$ (Corollary 4)

□

This lecture...

- Modern SAT solvers enormously successful in practice — key issue is to minimize time and memory consumption
- Modelled by proof size and space in proof complexity
- Simultaneous optimization of time and memory in proof systems gives us efficient protocols for search problems related to CNFs.
- Search problem for pebbling formulas for pyramids requires large communication!

. . . and next lecture

- more details about lifted formulas
- discuss $Search(Lift(F))$ vs $Lift(Search(F))$
- pebbling formulas for pyramids have large critical block sensitivity.
- comments on proof technique/limitations/open problems.