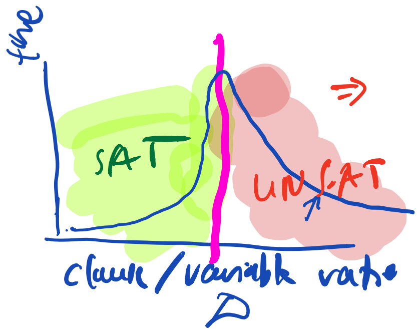
CSE 599S Proof Complexity

Lecture 7 21 October 2020

$\boxed{\text{Note: correction on Exercise 2}}$

Last time: n variable random 3-CNF formulas with $O(n)$ clauses are (an, c)-boundary expanders for constants $a, c > 0$ w.h.p.



$\Rightarrow$ For random 3-CNF formula F in n vars, $O(n)$ clauses w.h.p. Res(F) is $2^{\Omega(n)}$.

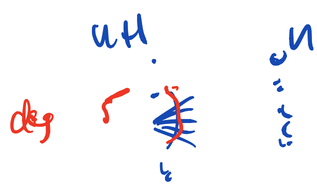
$2^{\Omega(n/\Delta)}$

Width-based l.b. quite general ✓ but ~~not~~ for PHP as is

$\sim n^2$ vars ✓ width $\sim n$ ✓

initial clause size $\sim n$ (pigeon clauses)

PHP(G) G bipartite graph of constant out-degree



random graph get an expander

not $\bigvee_{j \in [n]} x_{ij}$ wide $\bigvee_{(i,j) \in G} x_{ij}$ width 5 $O(n)$ edges

$$PHP^{n+1}_n = PHP(K_{n+1,n})$$ $K_{n+1,n}$ complete bipartite graph

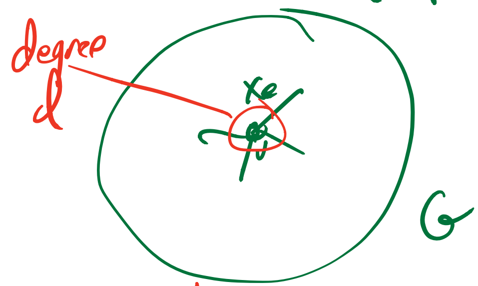
PHP(G) from PHP^{n+1}_n by setting some edges to 0.

width argument $\Rightarrow$ l.b. for PHP(G) $\Rightarrow$ l.b. for PHP^{n+1}_n

$TS(G, \ell)$ Tseitin formulas

undirected graph ↑ ↑ odd labelling

sum of 0,1 labels is odd.



2^{d-1} clauses

$$\sum_{e \text{ touches } v} x_e \equiv \ell(v) \bmod 2$$

eg If n is odd and all ℓ values are 1.

Handshake sum of degrees is even

Width lower bound ⇒ Tseitin formulas on expander graphs* require $2^{\Omega(n)}$ size resolution proofs.

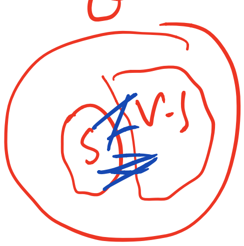
Parity reasoning

In practice need other tricks to deal with parity reasoning

$\boxed{\text{CryptoMiniSAT}}$

RAT proofs

Def^n An undirected graphs $G=(V,E)$ is an (r,c)-edge expander iff for all $S \subseteq V$, $|S| \le r$



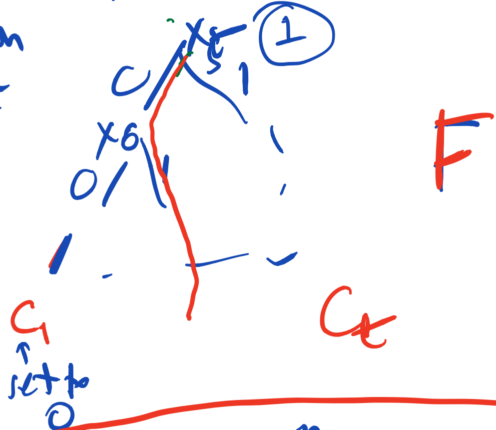
$$|E(S, V-S)| \ge c|S|$$

Thm $\exists$ (r,c)-edge expanders of degree 3

Characterization of Resolution Proofs

DPLL refutation Tree resolution

decision tree



$$F = \bigwedge_{i=1}^{m} C_i$$

Given $x \in \{0,1\}^n$ $F(x)=0$ since F is unSAT

Find i s.t. $C_i(x)=0$

Search problem for an unSAT F.

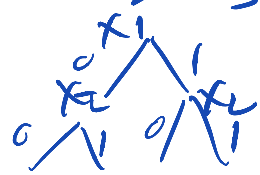
DPLL refutation of F $\equiv$ a decision tree solving search(F).

Subclass of Resolution	Model solving Search(F)
DPLL = Tree Resolution	Decision Tree
Ordered Resolution	OBDD
Regular Resolution	Read-once Branching Program

Decision Tree $\neq$ Decision D.A.G.

$x_1 \oplus x_2 \oplus x_3 \oplus x_4$

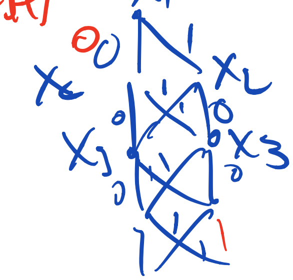
n vars



2^n leaves

sinks $\in [m]$

left 0 right 1



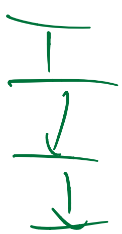
$2n+1$ nodes

Decision DAG $\equiv$ Binary Decision Diagram

$\equiv$ Branching Program

Special cases: • read-once Branching Program

- on any root-sink path any var is queried at most once.



layered • Oblivious read-once BP

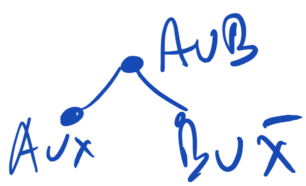
every path reads vars in same order

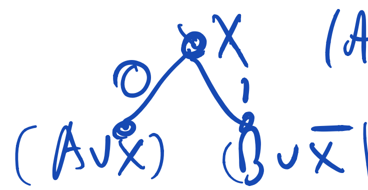
$\equiv$ Ordered Binary Decision Diagram

OBDD

Bryant -86

canonical form using FA minimization





$(A \cup B)$ falsify

But General Resolution for F $\not\equiv$ BP's for search(F)

could just test each clause one at time

Def A cube DAG protocol is a rooted d.a.g. s.t. each node v is labelled by a subcube of $\{0,1\}^n$

C_v = set of all inputs consistent with some partial asst. α_v

$C_u \subseteq C_v \cup C_w$

- $C_{root} = \{0,1\}^n$
- single output $i \in [m]$ — single label contains C_i

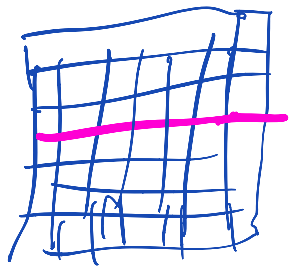
Claim $Res(F)$ = size of minimum cube dag protocol for Search(F)

clause C labelling

α_v partial asst to unique asst to variables of C making C false

Prover-Adversary game

Tseitin formula on grid graph



$\sqrt{n}$

$\sqrt{n}$

Our vert. width lower bound $\sqrt{n}$

$2^{\Omega(n)}$ resolution size

Dantchev-Riis

Prover wants to claim UNSAT

Adversary claims SAT.

Game: Prover maintain a partial asgt α.

initial $\emptyset$.

Round: • Prover asks adversary for the value of a var

• Adversary answers with a (possibly fake) value.

based only on α

- Prover forgets (erases from α) anything he doesn't need.

Game ends when prover has α that falsifies a clause of F.

Cost: # of possible configurations in the game.

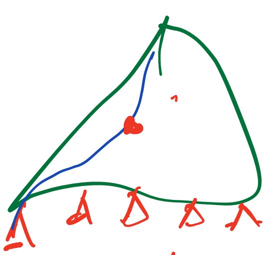
Fact: Tree Res, Order Res are incomparable (exponential gaps)

Regular Res vs Res

Separating examples: modified GT_n formulas

$\overline{x_{ij}} \vee \overline{x_{jk}} \vee x_{ik} \vee x_{f(i,j,k)}$

$\overline{x_{ij}} \vee x_{jk} \vee x_{ik} \vee \overline{x_{f(i,j,k)}}$



not regular

f maps triples to pairs in a funky way