

LECTURE 9

9: I

Do quick recap of size LB from last time
Space cplx in polynomial calculus

Only one paper [ABRW02]

Proves $\text{Space}(\text{PHP}_n^{m+1}) \geq n/4$.

Nothing else known for PCR or even PC

OPEN QUESTION 1

Are random k -CNFs formulas hard w.r.t space
for PCR? ~~For PC~~

Answer should be yes. Would be great to prove
this even for PC

The size of PHP_n^{n+1} is $\Theta(n^3)$, so expressed
in formula size N the best lower bound on
PCR-space is $\Omega(\sqrt[3]{N})$. (and this is tight).

Recall that worst-case upper bound known is $O(N)$

OPEN QUESTION 2

Are there (k) -CNF formulas F_n with
 $\text{Space}(F_n \vdash \perp) = \Omega(S(F_n))$?

One would expect random k -CNFs to
be an example here

An even more basic question has been open
Recall that we prefer working with k -CNFs

PCEP_n^m has width n .

Space lower bound in [ABRW02] much smaller: $n/4$

What about ~~monotone~~ k -CNF formulas?

Nothing known, not even for PC!
in particular, [ABRW02] breaks for $\widetilde{\text{PCEP}}_n^m$.

QUESTION 3

is it true that PCR, or even PC, can refute
any unsatisfiable k -CNF formula in
constant monomial space?

Answer "yes" has been consistent with
state of the art of current knowledge!
(But of course seemed absurd.)

Plan for this lecture was:

- ① Do [ABRW02] PCR-bound (this paper contains lots of other stuff as well)
- ② Discuss why it doesn't work for $\widetilde{\text{PCEP}}_n^m$ and what can be done
- ③ Discuss questions 1-3 more broadly

Except that as of ca 2 weeks, question 3
no longer open.

THEOREM 1 [FLNTZ '11]

There are unsatisfiable 4-CNF formulas F_n of size $\Theta(n)$ with $\text{SPR}(F_n \vdash \perp) = \Omega(\sqrt[3]{n})$.

We will do a slightly weaker result, which might be a bit easier to follow but which has all the needed ingredients for Thm 1.

THEOREM 2 [FLNTZ '11]

There are unsatisfiable CNF formulas F_n of size $\Theta(n)$ and width $\Theta(\log n)$ such that $\text{SPR}(F_n \vdash \perp) = \Omega(\sqrt[3]{n/\log n})$.

Thm 2 already big improvement of [ABRW02] Almost same space LB, but width now logarithmic instead of polynomial in n (so space LB exponential in width instead of $\leq$ width). Proofs very much inspired by [ABRW02], but with some twists.

Formulas in Thm 2 (and Thm 1) versions of PHP.

Basic PHP: Every pigeon has at least one hole (but can have many)

Functional PHP: Every pigeon has exactly one hole

Onto PHP: ... and every hole gets a pigeon

Onto-Functional PHP: Onto + Functional

Basic PHP_n^m $\bigvee_{j=1}^n x_{ij}$ $i \in [m]$
 $\bar{x}_{ij} \vee \bar{x}_{ij'}$ $i \neq i' \in [m]$
 $j \in [n]$

Functional axioms $\bar{x}_{ij} \vee \bar{x}_{ij'}$ $i \in [m]$
 $j \neq j' \in [n]$

Onto axioms $\bigvee_{i=1}^m x_{ij}$ $j \in [n]$

FPHP_n^m easier than PHP_n^m more axioms

Onto-PHP_n^m " PHP_n^m "

Onto-FPHP_n^m " FPHP_n^m and Onto-PHP_n^m "

Many papers study these different flavours of PHP to see what happens. Want to prove stronger LB for as strong formulas as possible.

We want to study functional PHP formulas but in a different encoding

Notational conventions

$$x^b = \begin{cases} x & \text{if } b=0 \\ \bar{x} & \text{if } b=1 \end{cases}$$

So $x^b = 0$, i.e., the literal x^b is true, iff $x=b$.

$$[n] = \{1, 2, \dots, n\}$$

$$[n, m) = \{n, n+1, \dots, m-1\}$$

DEF 3 (Bitwise Pigeonhole formula)

9: V

$$n = 2^L$$

Variables $x[p, i]$ = i th bit of number of hole p is sent to.
Hole $h = h_{L-1} h_{L-2} \dots h_1 h_0$ in binary

Think of holes as indexed from 0 to $n-1$

|| pigeons || 0 to $m-1$

The bitwise pigeonhole principle formula BPHP_n^m

has clauses for all pairs of pigeons $p_1, p_2 \in [0, m)$
 $p_1 \neq p_2$, and all $h \in [0, n)$ stating that
 p_1 & p_2 are not both sent to h

$$\bigvee_{i=0}^{L-1} x[p_1, i]^{1-h_i} \vee \bigvee_{i=0}^{L-1} x[p_2, i]^{1-h_i}$$

Note that any total assignment to $\text{Vars}(\text{BPHP}_n^m)$
defines a function from $[0, m)$ to $[0, n)$.

DEF 4 (Well-behaved assignment)

A total truth value assignment α to $\text{Vars}(\text{BPHP}_n^m)$
is well-behaved on a set of pigeons $S \subseteq [0, m)$
if α assigns all pigeons in S distinct holes.

Proof structure

Given formula F , space bound s

Want to prove $S_p(F \vdash \perp) \geq s$.

EQUIVALENTLY

Study any derivation $\pi = \{P_0 = \perp, P_1, \dots, P_L\}$

such that $S_p(\pi) < s$ and prove

that π doesn't derive contradiction.

(For PC/PCR, that $1 \neq P_L$).

That is, prove that P_i is satisfiable for all i .

How? We don't know much about structure of P_i .

Even for resolution this is hard, and for PC/PCR polynomials can start to cancel and strange things happen...

Idea Construct "auxiliary configurations" A_i

that tell us sth about P_i but which are easier to understand

- ① A_i implies P_i (stronger)
- ② A_i satisfiable (so P_i is too)
- ③ For derivation step $P_i \rightarrow P_{i+1}$
can update A_i to A_{i+1} as long
as $S_p(P_i) < s$.

The challenge is to get all the details right so that this idea works. We will look at implications w.r.t well-behaved assignments (not all total assignments)

Let us next look at the "easier to understand" auxiliary configurations. 9: VII

We think of these configs as "committing" to putting certain pigeons in certain ~~holes~~ halves of the holes (by fixing bits $x[p_i, i]$).

DEF 5 (Commitment)

A (disjunctive) commitment C is a 2-clause $C = x[p_1, i_1]^{b_1} \vee x[p_2, i_2]^{b_2}$ where $p_1 \neq p_2$

Let $\text{dom}(C) = \{p_1, p_2\}$

A commitment set $\mathcal{A}$ is a set of commitments $\mathcal{A} = \{C_1, C_2, \dots, C_s\}$

where $\text{dom}(C_i) \cap \text{dom}(C_j) = \emptyset$ if $i \neq j$.

The size of $\mathcal{A}$ is $|\mathcal{A}| = s = \text{dom}(\mathcal{A})/2$

($\text{dom}(\mathcal{A}) = \bigcup_{C \in \mathcal{A}} \text{dom}(C)$).

α is well-behaved on and satisfies $\mathcal{A}$ if

- (1) α well-behaved on $\text{dom}(\mathcal{A})$
- (2) α satisfies $\mathcal{A}$

[Note that α satisfies $x[p_1, i_1]^{b_1} \vee x[p_2, i_2]^{b_2}$ if $x[p_j, i_j] = b_j$ for $j=1$ or $j=2$]

LEMMA 6

Suppose $S \subseteq [0, m)$, $|S| < n/2$

α well-behaved on S

$x[p, i]^b$ literal associated with pigeon $p \notin S$.

Then can modify α to β by reassigning p [i.e., changing values of $\alpha[p, j]$ ^{vars}]
so that

- β is well-behaved on $S \cup \{p\}$
- β satisfies $x[p, i]^b$

Proof There are $n/2$ holes for p whose i th bit is b . Pigeons in S use $< n/2$ holes. Hence $\exists$ hole h not assigned to any pigeon in S such that p can be sent to h and $x[p, i]^b$ is satisfied.

COROLLARY 7 $\subseteq [0, m)$

Let S, T be ~~disjoint~~ set of pigeons s.t.

$$S \cap T = \emptyset$$

$$|S \cup T| \leq n/2$$

and let $\mathcal{X}$ be set containing exactly one literal $x[p, i]^{b_p}$ for every $p \in T$.

Then any α well-behaved on S can be modified to β by reassigning pigeons in T so that

- β is well-behaved on $S \cup T$
- β satisfies all literals in $\mathcal{X}$.

Proof of Cor 7

Consider pigeons in T one by one and apply Lemma 6.

Now, finally, let us define what it means that a commitment set implies, or entails a set of multilinear polynomials.

DEF 8 (Entailment)

A commitment set A entails a PCR-config $\mathbb{P}$ over well-behaved assignments if every (total) assignment α which is well-behaved on A and satisfies A must also satisfy $\mathbb{P}$.

[That is, for every $p \in \mathbb{P}$, $p(\alpha) = 0$]

The key to the lower bound is

LEMMA 9 Localizing lemma / Magic hammer

Suppose A commitment set, $|A| \leq n/4$
 $\mathbb{P}$ PCR-config

A entails $\mathbb{P}$ over well-behaved assignments

Then there is a commitment set B of size $|B| \leq 2 \cdot \text{Sp}(\mathbb{P})$ such that B entails $\mathbb{P}$ over well-behaved assignments.

This is exactly what we need to make our proof idea go through.

9: I

This lemma is similar to a Locality lemma in [ABRW02] and our proof follows the same general outline as theirs. However, we need to do things a bit differently to make the proof work (for the BPPEP formulas and for the 4-CNF formulas we won't talk about today).

Next time we will show:

- How Thm 2 follows from the Locality lemma and the other lemmas proven so far
- How to prove the Locality lemma, which is the heart of the whole argument.