CS151 Complexity Theory

Lecture 6
April 20, 2023

1

Relation to other classes

- Clearly **NC** ⊆ **P**
  - recall **P** ≡ uniform poly-size circuits
- **NC$_1$ ⊆ L**
  - on input x, compose logspace algorithms for:
    - generating $C_{|x|}$
    - converting to formula
    - FVAL

April 20, 2023 CS151 Lecture 6 2

2

Relation to other classes

- **NL** ⊆ **NC$_2$**: S-T-CONN ∈ **NC$_2$**
  - given G = (V, E), vertices s, t
  - A = adjacency matrix (with self-loops)
  - $(A^2)_{i,j}$ = 1 iff path of length ≤ 2 from node i to node j
  - $(A^n)_{i,j}$ = 1 iff path of length ≤ n from node i to node j
  - compute with depth log n tree of Boolean matrix multiplications, output entry s, t
  - $\log^2$ n depth total

April 20, 2023 CS151 Lecture 6 3

3

NC vs. **P**

- can every efficient algorithm be efficiently parallelized?

$$\mathbf{NC} \stackrel{?}{=} \mathbf{P}$$

- **P**-complete problems least-likely to be parallelizable
  - if **P**-complete problem is in **NC**, then **P** = **NC**
  - Why?
    we use logspace reductions to show problem **P**-complete; **L** in **NC**

April 20, 2023 CS151 Lecture 6 4

4

NC vs. **P**

- can every uniform, poly-size Boolean circuit family be converted into a uniform, poly-size Boolean formula family?

$$\mathbf{NC_1} \stackrel{?}{=} \mathbf{P}$$

April 20, 2023 CS151 Lecture 6 5

5

NC Hierarchy Collapse

$$NC_1 \subseteq NC_2 \subseteq NC_3 \subseteq NC_4 \subseteq \ldots \subseteq NC$$

Exercise

if $NC_i = NC_{i+1}$, then $NC = NC_i$

(prove for non-uniform versions of classes)

April 20, 2023 CS151 Lecture 6 6

6

Lower bounds

- Recall: "**NP** does not have polynomial-size circuits" (**NP** $\subsetneq$ **P/poly**) implies **P** $\neq$ **NP**
- **major goal**: prove lower bounds on (non-uniform) circuit size for problems in **NP**
  - believe exponential
  - super-polynomial enough for **P** $\neq$ **NP**
  - best bound known: $(5-o(1))\cdot n$
  - don't even have super-polynomial bounds for problems in **NEXP**

April 20, 2023 CS151 Lecture 6 7

7

Lower bounds

- lots of work on lower bounds for restricted classes of circuits
  - we'll see two such lower bounds:
    - formulas
    - monotone circuits

April 20, 2023 CS151 Lecture 6 8

8

Shannon's counting argument

- frustrating fact: *almost all* functions require huge circuits

Theorem (Shannon): With probability at least $1 - o(1)$, a random function

$$f:\{0,1\}^n \rightarrow \{0,1\}$$

requires a circuit of size $\Omega(2^n/n)$.

April 20, 2023 CS151 Lecture 6 9

9

Shannon's counting argument

- Proof (counting):
  - $B(n) = 2^{2^n}$ = # functions $f:\{0,1\}^n \rightarrow \{0,1\}$
  - # circuits with n inputs + size s, is at most

$$C(n, s) \leq ((n+3)s^2)^s$$

s gates

n+3 gate types

2 inputs per gate

April 20, 2023 CS151 Lecture 6 10

10

Shannon's counting argument

$$\boxed{C(n, s) \leq ((n+3)s^2)^s}$$

- $C(n, c2^n/n) < ((2n)c^2 2^{2n}/n^2)^{(c2^n/n)}$

  $< o(1)2^{2c2^n}$

  $< o(1)2^{2^n}$ (if $c \leq ½$)

- probability a random function has a circuit of size $s = (½)2^n/n$ is at most

$$C(n, s)/B(n) < o(1)$$

April 20, 2023 CS151 Lecture 6 11

11

Shannon's counting argument

- frustrating fact: *almost all* functions require huge **formulas**

Theorem (Shannon): With probability at least $1 - o(1)$, a random function

$$f:\{0,1\}^n \rightarrow \{0,1\}$$

requires a **formula** of size $\Omega(2^n/\log n)$.

April 20, 2023 CS151 Lecture 6 12

12

Shannon's counting argument

- Proof (counting):
  - $B(n) = 2^{2^n}$ = # functions $f:\{0,1\}^n \to \{0,1\}$
  - # formulas with n inputs + size s, is at most

$$F(n, s) \le 4^s 2^s (2n)^s$$

4^s binary trees with s internal nodes

2 gate choices per internal node

2n choices per leaf

Shannon's counting argument

$F(n, s) \le 4^s 2^s (2n)^s$

- $F(n, c2^n/\log n) < (16n)^{(c2^n/\log n)}$
  $< 16^{(c2^n/\log n)} 2^{(c2^n)} = (1 + o(1))2^{(c2^n)}$
  $< o(1)2^{2^n}$ (if $c \le ½$)
- probability a random function has a **formula** of size $s = (½)2^n/\log n$ is at most $F(n, s)/B(n) < o(1)$

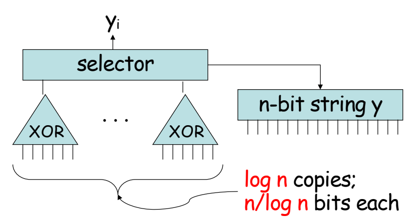
Andreev function

- best formula lower bound for language in NP:

Theorem (Andreev, Hastad '93): the Andreev function requires $(\wedge,\vee,\neg)$-formulas of size at least

$$\Omega(n^{3-o(1)}).$$

Andreev function



the Andreev function A(x,y)

$A:\{0,1\}^{2n} \to \{0,1\}$

Random restrictions

- key idea: given function

  $f:\{0,1\}^n \to \{0,1\}$

  restrict by ρ to get f_ρ
  - ρ sets some variables to 0/1, others remain free
- $R(n, \epsilon n)$ = set of restrictions that leave ϵn variables free
- Definition: $L(f)$ = smallest $(\wedge,\vee,\neg)$ formula computing f (measured as leaf-size)

Random restrictions

- observation:

$$E_{\rho\leftarrow R(n,\,\epsilon n)}[L(f_\rho)] \le \epsilon L(f)$$

  - each leaf survives with probability ϵ
- may shrink more…
  - propogate constants

Lemma (Hastad 93): for all f

$$E_{\rho\leftarrow R(n,\,\epsilon n)}[L(f_\rho)] \le O(\epsilon^{2-o(1)} L(f))$$

Hastad's shrinkage result

- Proof of theorem:
  - Recall: there exists a function

    $h:\{0,1\}^{\log n} \rightarrow \{0,1\}$

    for which $L(h) > n/2\log\log n$.
  - hardwire truth table of that function into y to get $A^*(x)$
  - apply random restriction from

    $R(n, m = 2(\log n)(\ln \log n))$

    to $A^*(x)$.

April 20, 2023 CS151 Lecture 6 19

19

The lower bound

- Proof of theorem (continued):
  - probability given XOR is killed by restriction is probability that we "miss it" m times:

    $(1 - (n/\log n)/n)^m \leq (1 - 1/\log n)^m$

    $\leq (1/e)^{2\ln \log n} \leq 1/\log^2 n$
  - probability even one of XORs is killed by restriction is at most:

    $\log n(1/\log^2 n) = 1/\log n < ½.$

April 20, 2023 CS151 Lecture 6 20

20

The lower bound

- (1): probability even one of XORs is killed by restriction is at most:

  $\log n(1/\log^2 n) = 1/\log n < ½.$
- (2): by Markov:

  $\Pr[\, L(A^*_\rho) > 2\, E_{\rho \leftarrow R(n,\, m)}[L(A^*_\rho)] \,] < ½.$
- Conclude: for *some* restriction ρ
  - all XORs survive, and
  - $L(A^*_\rho) \leq 2\, E_{\rho \leftarrow R(n,\, m)}\, [L(A^*_\rho)]$

April 20, 2023 CS151 Lecture 6 21

21

The lower bound

- Proof of theorem (continued):
  - if all XORs survive, can restrict formula further to compute hard function h
    - may need to add ¬'s

    $L(h) = \mathbf{n}/2\log\log n \leq L(A^*_\rho)$

    $\leq 2E_{\rho \leftarrow R(n,\, m)}[L(A^*_\rho)] \leq O((m/n)^{2-o(1)} L(A^*))$

    $\leq O(\, ((\log n)(\ln \log n)/\mathbf{n})^{\mathbf{2-o(1)}}\, L(A^*)\,)$
  - implies $\Omega(n^{3-o(1)}) \leq L(A^*) \leq L(A).$

April 20, 2023 CS151 Lecture 6 22

22

Clique

CLIQUE = { (G, k) | G is a graph with a clique of size ≥ k }

(clique = set of vertices every pair of which are connected by an edge)

- CLIQUE is **NP**-complete.

April 20, 2023 CS151 Lecture 6 23

23

Circuit lower bounds

- We think that **NP** requires exponential-size circuits.
- Where should we look for a problem to attempt to prove this?
- Intuition: "hardest problems" – i.e., **NP-complete problems**

April 20, 2023 CS151 Lecture 6 24

24

Circuit lower bounds

- Formally:
  - if *any* problem in **NP** requires super-polynomial size circuits
  - then *every* **NP**-complete problem requires super-polynomial size circuits
  - Proof idea: poly-time reductions can be performed by poly-size circuits using a variant of CVAL construction

25

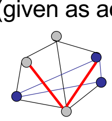
Monotone problems

- Definition: monotone language = language $L \subseteq \{0,1\}^*$ such that $x \in L$ implies $x' \in L$ for all $x \preccurlyeq x'$.
  - flipping a bit of the input from 0 to 1 can only change the output from "no" to "yes" (or not at all)

26

Monotone problems

- some **NP**-complete languages are monotone
  - e.g. CLIQUE (given as adjacency matrix):



  - others: HAMILTON CYCLE, SET COVER…
  - but not SAT, KNAPSACK…

27

Monotone circuits

A restricted class of circuits:

- Definition: monotone circuit = circuit whose gates are ANDs (∧), ORs (∨), but no NOTs
- can compute exactly the monotone fns.
  - monotone functions closed under AND, OR

28

Monotone circuits

- A question:

Do all poly-time computable monotone functions have poly-size monotone circuits?

  - recall: true in non-monotone case

29

Monotone circuits

A monotone circuit for $\text{CLIQUE}_{n,k}$

- Input: graph $G = (V,E)$ as adj. matrix, $|V|=n$
  - variable $x_{i,j}$ for each possible edge (i,j)
- ISCLIQUE(S) = monotone circuit that = 1 iff $S \subseteq V$ is a clique: $\bigwedge_{i,j \in S} x_{i,j}$

$\text{CLIQUE}_{n,k}$ computed by monotone circuit:

$$\bigvee_{S \subseteq V, |S|=k} \text{ISCLIQUE(S)}$$

30

Monotone circuits

- Size of this monotone circuit for $CLIQUE_{n,k}$: $\binom{n}{k}\binom{k}{2}$
- when $k = n^{1/4}$, size is approximately:

$$\left(\frac{n}{n^{1/4}}\right)^{n^{1/4}} \left(\frac{n^{1/4}}{2}\right)^{2} \approx n^{\Omega(n^{1/4})}$$

31

Monotone circuits

- Theorem (Razborov 85): monotone circuits for $CLIQUE_{n,k}$ with $k = n^{1/4}$ must have size at least

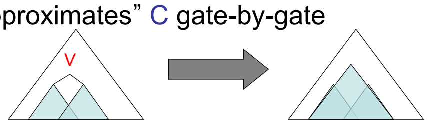
$$2^{\Omega(n^{1/8})}.$$

- Proof:
  - rest of lecture

32

Proof idea

- "method of approximation"
- suppose C is a monotone circuit for $CLIQUE_{n,k}$
- build another monotone circuit CC that "approximates" C gate-by-gate



33

Proof idea

- on test collection of positive/negative instances of $CLIQUE_{n,k}$:
  - local property: few errors at each gate
  - global property: many errors on test collection
- Conclude: C has many gates

34

Notation

- input: graph G = (V, E)
- variable $x_{j,k}$ for each potential edge (j, k)
- $CC(X_1, X_2, \ldots X_m)$, where $X_i \subseteq V$, means:

$$\bigvee_i \left(\bigwedge_{j,k \in X_i} x_{j,k}\right) *$$

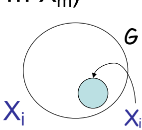
- For example: $CC(X_1, X_2, \ldots X_m)$ where the X_i range over all k-subsets of V
  - this is the obvious monotone circuit for $CLIQUE_{n,k}$ from a previous slide.

*[CC( ) = 0; $(\bigwedge_{i,j \in \emptyset} x_{i,j}) = 1$]

35

Preview

- approximate circuit $CC(X_1, X_2, \ldots X_m)$
- n = # nodes
- $k = n^{1/4}$ = size of clique
- $h = n^{1/8}$ = max. size of subsets X_i
  - this is "global property" that ensures lots of errors
  - many graphs G with no k-cliques, but clique on X_i of size h



36

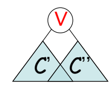
Preview

- approximate circuit $CC(X_1, X_2, \ldots X_m)$
- $p = n^{1/8} \log n$
- $M = (p - 1)^h h!$
- max # of subsets is M (so $m \le M$)
  - critical for "local property" that ensures few errors at each gate

37

Building CC

- CC ("crude circuit") for circuit C defined inductively as follows:
  - CC for single variable $x_{j,k}$ is just CC({ j, k })
    - no errors yet!
  - CC for circuit C of form:

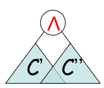


  - "approximate OR" of CC for C', CC for C''

38

Building CC

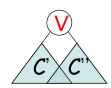
- CC for circuit C of form:



- "approximate AND" of CC for C', CC for C''
- "approximate OR" and "approximate AND" steps introduce errors

39

Approximate OR



$CC(X_1,X_2,\ldots X_{m'})$ $CC(Y_1,Y_2,\ldots Y_{m''})$

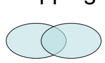
- exact OR:

  $CC(X_1,X_2,\ldots X_{m'},Y_1,Y_2,\ldots Y_{m''})$
  - set sizes still $\le h$
  - may be up to 2M sets; need to reduce to M

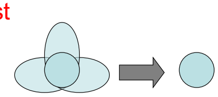
40

Approximate OR

- throw away sets? bad:many errors
- throw away overlapping sets? – better



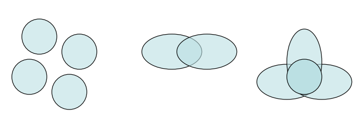
- throw away special configuration of overlapping sets – best



41

Sunflowers

- Definition: (h, p)-sunflower is a family of p sets, each of size at most h, such that intersection of every pair is a subset S (the "core").



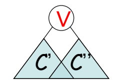
42

Sunflowers

Lemma (Erdös-Rado): Every family of more than $M = (p\text{-}1)^h h!$ sets, each of size at most h, contains an (h, p)-sunflower.

- Proof:
  - not hard
  - in Papadimitriou, elsewhere

Approximate OR

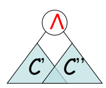


- $CC(X_1,X_2,\dots X_{m'})$
- $CC(Y_1,Y_2,\dots Y_{m''})$
- exact OR:

  $CC(X_1,X_2,\dots X_{m'},Y_1,Y_2,\dots Y_{m''})$
  - while more than M sets, find (h, p)-sunflower; replace with its core ("**pluck**")
- approximate OR:

  $CC(\mathbf{pluck}(X_1,X_2,\dots X_{m'},Y_1,Y_2,\dots Y_{m''}))$

Approximate AND



- $CC(X_1,X_2,\dots X_{m'})$
- $CC(Y_1,Y_2,\dots Y_{m''})$
- (close to) exact AND:

  $CC(\{(X_i \cup Y_j) : 1 \le i \le m', 1 \le j \le m''\})$
  - some sets may be larger than h; discard them
  - may be up to M^2 sets. While > M sets, find (h, p)-sunflower; replace with its core ("**pluck**")
- approximate AND:

  $CC(\mathbf{pluck}(\{(X_i \cup Y_j) : |X_i \cup Y_j| \le h\}))$