Lower Bounds **(CS294-152, Fall'18)**

Nonuniform lower bounds on functions in P

Notes for 9/10/18

Chinmay Nirkhe

Announcements:

Simons workshop this week! Students enrolled: please send your "review" of a talk by next Monday.

1 Preliminaries

Last time we saw time-space lower bounds for SAT, and we saw a bit about relativization and oracles, in particular an oracle relative to which the SAT lower bound is false. Of course SAT is an NP-complete problem... what about decision problems in P? Lower bounds for these have mainly been studied in a *non-uniform* model of space-bounded computation.

2 Branching Programs

Definition 2.1 (Branching Program [BC82])**.** *Let Σ be a finite alphabet and $n > 0$. A $|\Sigma|$-way* branching program *of length $\ell(n)$ and size $s(n)$ is a function computed on input $x \in \Sigma^n$. It is defined by a $s(n)$ size directed acyclic graph with a start node r with the length of all paths $\leq \ell(n)$. Each node u is labeled with an index $i_u \in [n]$ and the out-degree of each node is $|\Sigma|$ with one edge labeled with each element of Σ. Each node u has an output letter $o_u \in \Sigma \cup \{\varepsilon\}$.*

Computation is defined inductively. Let $v_1 = r$. v_{j+1} is inductively selected as the vertex as the edge reached from v_j after following the edge with label $x_{i_{v_j}} \in \Sigma$. This creates a path $v_1 \dots v_t$ for $t \leq \ell(n)$ and the output is $o_{v_1} \dots o_{v_t}$.

Computation on a branching program runs in additional space $\log(s(n))$ and requires time $\ell(n)\log(s(n))$. The space is due to storing the current node and the time due to the restriction on path length.

In addition, these models are perfect for capturing the notion of Random Access Machines (RAMs) including word RAMs for words of $O(1)$ size.

Theorem 2.1. *Let $f : \Sigma^\star \to \Sigma^\star$ be computed by a RAM TM in time $t(n) \geq n$ and size $s(n) \geq \log n$. Then $f_n : \Sigma^n \to \Sigma^*$ (the restriction of f to inputs of length n) has a $|\Sigma|$-way branching program of length $t(n)$ and size $2^{O(s(n))}$.*

Proof. Let M be the RAM TM. Construct a node for each of the $2^{O(s)}$ configurations of the memory of M. For nodes $(\mathbf{mem}_1, i)$ and $(\mathbf{mem}_2, j)$ draw an edge $(\mathbf{mem}_1, i) \to (\mathbf{mem}_2, j)$ with the label σ if $x_i = \sigma$ would transition the head to bit x_j and memory $\mathbf{mem}_2$. The starting configuration r is the initial state and halting states are leaves[1]. By the bound on the computation, this branching program has depth t. □

If Σ can depend on n, then this can also capture the "word-RAM" model in which the input is stored in words of, say, $\log(n)$ bits, and we can output words of $\log(n)$ bits.

3 Decision Problems in P

Next we explore the best known branching program lower bounds known for decision problems in P.

Theorem 3.1 (Simplified [Ajt99, BSSV03])**.** *There are problems on Boolean input solvable in $O(nlogn)$ time requiring branching program (families) of $\Omega(n(logn)^\delta)$ length to solve with $n^{.999}$ space – i.e. $2^{n^{.999}}$ size for various $\delta \leq 1$ But when the space is lowered to e.g. $O(logn)$, the length lower bound is still around $n(logn)^{1/2}$.*

Roughly, these problems are decision versions of *vector convolutions*: given vectors $x, y \in \mathbb{F}_2^n$, for all $i = 0, \ldots, n-1$, compute

$$z = \sum_{j=0}^{n-1} x_j \cdot y_{(i+j \pmod n)} \tag{1}$$

and then output $\langle x, z \rangle$.

[Ryan's aside] Is $\mathsf{TIME}[n]$ contained in $\mathsf{TS}[n^{1.1}, O(\log n)]$ or even is $\mathsf{TIME}[n]$ contained in $\mathsf{TS}[O(n), O(\log n)]$?

[1] We know it is a DAG – i.e. has no cycles – because this would imply the TM doesn't halt.

4 Function Problems

It is easier to prove lower bounds in the setting of function problems (which output many bits instead of one bits). In fact, the result we will prove critically relies on the fact that the function outputs many bits.

Let's consider a family of functions $f_n : \Sigma^n \to \Sigma^n$ where Σ is a finite alphabet (that may depend on n).

Theorem 4.1 ([BC82])**.** *Sorting n integers from the range $[n, n^2]$ requires time-space product $\Omega(n^2/\log n)$.*

By Theorem 2.1, we can instead show that any branching program for sorting n integers from the range $[n, n^2]$ requires

$$\text{time} \times \log(\text{space}) \geq \Omega(n^2/\log n). \tag{2}$$

A result of Beame [Bea91] shows that an upperbound of $O(n^2)$ for time $\times \log(\text{space})$.

This result is harder than the one we will prove in this lecture. Instead we will give a $\Omega(n^2)$ lower bound for a problem that is simpler than sorting (it implies lower bounds for sorting).

Problem 4.1 (Non-Occuring Elements (NOE))**.** *Given a list L of n elements from $[n]$ (with possible repetition), print the nonoccuring elements, $[n] - L$ in any order.*

Proposition 4.1. *For any $n \leq T(n) \leq n^2/(\log n)$ and $S(n) \geq \log n$, NOE can be solved on a $(\log n)$-word RAM TM in time $T(n)$ and space $S(n)$ where $T(n)S(n) \leq O(n^2)$.*

The idea is to partitition the set $[n]$ into n/S blocks, each of S items. Then, we perform n/S passes over the input list; in the i-th pass, determine which items in the i-th block are not occuring in the list.

This will take space $O(S)$ and time per pass of $O(n)$ for a total time of $O(n^2/S)$.

This proves the timespace upperbound. The lowerbound is due to Beame [Bea91]. Due to Theorem 2.1, the following result proves the optimality of the previous algorithm, even for the $(\log n)$-word RAM.

Theorem 4.2 (Adaptation of [Bea91])**.** *For every n-way branching program for NOE with $T(n)$ length and size $2^{S(n)}$, $T(n) \cdot S(n) \geq \Omega(n^2)$.*

There are two main ingredients to this proof.

Step 1: Uniform random inputs to NOE require a long output

Proposition 4.2. *There is a $\delta > 0$ such that for sufficiently large n,*

$$\Pr_{L \in_R [n]^n} [L \textit{ contains at least } \delta n \textit{ nonoccuring elements}] \geq \delta. \tag{3}$$

Proof. (Sketch) Balls-and-bins argument. When you thorw n balls into n bins, what's the probability that least δn of the bins are empty? Let Z be a r.v. indicating the number of empty bins. Goal is to find a $\delta \geq 0$ such that $\Pr[Z \geq \delta n] \geq \delta$.

For any index i,

$$\Pr[i \notin L] = \left(1 - \frac{1}{n}\right)^n \to \frac{1}{e} \tag{4}$$

By linearity, $E[Z] \approx n/e$ for large n. Then by method of bounded differences, for all t,

$$\Pr\left[|Z - E[Z]| > t\right] \leq 2 \exp\left(\frac{-2t^2}{n}\right). \tag{5}$$

Choose $t = \delta n$. Then

$$\Pr[Z \geq \frac{n}{e} - \delta n] \geq 1 - 2\exp\left(-2\delta^2 n\right). \tag{6}$$

For a choice of $\delta \in (1/e, 1)$, this probability will be greater than δ for sufficiently large n. □

Step 2: All short branching programs have low probability of printing many outputs of a random input

Lemma 4.1. *For all branching programs P of length $\leq n/2$ which make m outputs for $m \leq n/2$,*

$$\Pr_{L \in_R [n]^n} [P \textit{ outputs } m \textit{ NOEs of } L] \leq \exp(-m/2). \tag{7}$$

Proof. Let π be a full path in the branching program (from start to sink). Then an equivalent statement to the lemma is

$$\sum_{\pi} \Pr_L [P \text{ follows } \pi \text{ on } L] \cdot \Pr_L [P \text{ outputs } m \text{ NOEs of } L | P \text{ follows } \pi \text{ on } L] \leq \exp\left(\frac{-m}{2}\right). \tag{8}$$

It suffices then to show that for any π,

$$\Pr_L [P \text{ outputs } m \text{ NOEs of } L | P \text{ follows } \pi \text{ on } L] \leq \exp\left(\frac{-m}{2}\right) \tag{9}$$

as $\sum_\pi \Pr_L [P \text{ follows } \pi \text{ on } L] \leq 1$.

To prove (9), notice that

$$\text{LHS of (9)} = \frac{\#\text{ lists } \in [n]^n \text{ consistent with } \pi \text{ and produces } m \text{ outputs of } P}{\#\text{ lists } \in [n]^n \text{ consistent with path } \pi}. \tag{10}$$

This will ammount to lower bounding the denominator and upper bounding the numberator. Let $q \leq n/2$ be the number of distinct input variables queried along the path π. Then the denominator is n^{n-q} as there are that many possible lists that are consistent with π (b.c. there are $n-q$ unread and there are n possibilities for each of those unread variables.)

For the numerator, all $n-q$ variables not queried in π must *not* take any values among the m outputs because the m outputs are the non-occuring elements. Then, there are at most $(n-,)^{n-q}$ possible lists consistent with π for which all m outputs of P are correct.

Then,

$$\begin{aligned}
\text{LHS of (9)} &= \frac{(n-m)^{n-q}}{n^{n-q}} && (11)\\
&\leq \left(1 - \frac{m}{n}\right)^{n/2} && (12)\\
&\leq \exp\left(-\frac{n}{2} \cdot \frac{m}{n}\right) && (13)\\
&\leq \exp(-m/2). && (14)
\end{aligned}$$

We used $1 - x \leq \exp(-x)$ at the end.

□

Now we can prove Theorem 4.2.

Proof of Theorem 4.2. Assume P is a branching program for NOE with length $T = T(n)$ and size 2^S for $S = S(n)$. We show that if $T \cdot S \leq \alpha n^2$ for a certain $\alpha > 0$, then there exists an input on which P is incorrect.

The first step is to layer P. We force P to have T layers and each layer has $O(2^S)$ nodes with the starting node in the first layer and every edge from a node in layer i goes to layer $i+1$. This is achievable with repeating some nodes (this can blow up the size to $O(T2^S)$ but it has the same length).

Consider a random list $L \in_R [n]^n$ as input to P. By Step 1, there exists a $\delta > 0$ such that

$$\Pr_L [\#\text{ of NOE for } L \text{ is } \geq \delta n] \geq \delta. \tag{15}$$

We now clain that if $TS < \frac{\delta}{10}n^2$, then

$$\Pr_L [P \text{ outputs } \geq \delta n \text{ NOE of } L] \ll \delta. \tag{16}$$

Then, there is some L such that P is wrong on L. How does one prove (16)? We reduce the problem to one about short branching programs.

Consider the event that P outputs $\geq \delta n$ NOEs of L. We can partition P into p "parts" for $p = 2T/n$, each part consisting of $n/2$ layers[2]. When P outputs $\geq \delta n$ NOEs of L, then some part[3] outputs $\geq \delta n/p = \delta n^2/(2T)$ NOEs of L. Let this part be the "important" parts of L (the exact part depends on the input L).

$$\Pr_L[P \text{ outputs } \geq \delta n \text{ NOEs of } L] \leq \Pr_L \left[\text{important part outputs } \geq \frac{\delta n^2}{2T} \text{ NOEs}\right]. \tag{17}$$

By Step 2, every length $n/2$ branching program P' with outputs $m = \delta n^2/(2T)$,

$$\Pr_L [P' \text{ outputs } m \text{ NOEs of } L] \leq \exp(-m/2) = \exp\left(\frac{-\delta n^2}{4T}\right). \Pr_L [P' \text{ outputs } m \text{ NOEs of } L] \leq \exp(-m/2) = \epsilon \tag{18}$$

However, we don't know on a random input L within the important part will be taken, so we take a lazy union bound over all 2^S nodes that could start the important part. By the upper bound assumption on TS we have $S \leq \delta n^2/10$:

$$\Pr_L \left[\text{important part on } L \text{ outputs } \geq \frac{\delta n^2}{2T} \text{ NOEs}\right] \leq 2^S \cdot \exp\left(\frac{-\delta n^2}{4T}\right) \tag{19}$$

$$\leq \exp\left(\frac{\delta n^2}{10} \cdot \frac{-\delta n^2}{4T}\right) \tag{20}$$

$$\leq \exp\left(-\frac{\delta^2 n^4}{40T}\right) \tag{21}$$

This probability goes to 0 as n increases since $T \leq O(n^2)$. For sufficiently large n, this is $\ll \delta$. This proves (16). □

Notice why the function problem was necessary. A fundamental part of the argument was about arguing that branching programs must output or can't output too many values. Its hard to see how to change that to decision problems in any easy way.

[2]Think of each part as being a collection of $n/2$-length "subprograms".

[3]Maybe one could call this a hybrid argument flavored.

References

[Ajt99] Miklós Ajtai. A non-linear time lower bound for boolean branching programs. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 60–70, 1999.

[BC82] A. Borodin and S. Cook. A time-space tradeoff for sorting on a general sequential model of computation. *SIAM Journal on Computing*, 11(2):287–297, 1982.

[Bea91] Paul Beame. A general sequential time-space tradeoff for finding unique elements. *SIAM J. Comput.*, 20(2):270–277, 1991.

[BSSV03] Paul Beame, Michael E. Saks, Xiaodong Sun, and Erik Vee. Time-space trade-off lower bounds for randomized computation of decision problems. *J. ACM*, 50(2):154–195, 2003.