Guest Lecture: Average Case depth hierarchy for $\wedge, \vee, \neg$-Boolean circuits.

Boaz Barak

Thanks to the authors of RST for answering my questions by email and in particular to Li-Yang Tan for sharing with me his slides.

Today we are going to cover the following recent result of Rossman, Servedio and Tan:

Theorem 1. *For every d there exists a function F computable by a poly-size depth d circuit (with $\wedge, \vee$ and $\neg$ gates) such that for every $\exp(n^{o(1)})$-size depth $d-1$ circuit C,*

$$\Pr_{x \in \{0,1\}^n}[C(x) = F(x)] \leq 1/2 + o(1)$$

(The quantitative parameters are stronger — see the paper.)

As they show, the result immediately resolves two questions considered by prior researchers:

1. It confirms a 1986 conjecture of Håstad (also raised by and discussed by many others) that the polynomial hierarchy is infinite with respect to a random oracle. (We've already had strong reasons to believe Håstad's conjecture, since Book showed in 1994 that it's true unless the polynomial hierarchy collapses.) The prior best result was by O'Donnell and Wimmer in 2007, who showed that $\Sigma_3 \not\subseteq \Sigma_2$ w.r.t. a random oracle. However, they did not state their result in this language, and it's unclear if they were aware of this connection. (Aaronson 2010, showed the (incomparable?) result that $\Pi_2 \not\subseteq P^{NP}$ with respect to a random oracle, and also realized the connections between these questions and questions on the analysis of Boolean functions, though not the relation to the work of O'Donnell and Wimmer.)

2. It refutes even very weak variants of a conjecture of Benjamini, Kalai and Schramm (BKS) from 1999 that low influence functions can be approximated by circuits of small depth. (This is in the regime of *logarithmic* influence; Friedgut's theorem says that functions of *constant* influence are juntas and hence no longer applies in this regime but one still would like to derive some structural information on the function based on its low influence, given that the canonical examples all have low depth.) The original conjecture was refuted by O'Donnell and Wimmer in the 2007 work mentioned above, but O'Donnell, Kalai, and Hatami raised the question of whether some weaker variants still hold, which are now refuted by this new result. (Given that the new example is ultimately a low depth circuit, there may still be interesting versions of this conjecture that might be true.)

Why do we care about random oracles?

The central results of computability theory all were based on treating Turing machines as a *black box.* For example, one shows that there is an uncomputable problem by defining the function that maps any Turing machine M to the negation of $M(M)$. All the properties of Turing Machines that those results required are that one can encode a machine as a string, and that there is a *universal* Turing machine that given an encoding of a TM M and an encoding of a string x, can simulate M's execution on the input x. For every function (known as an *oracle*) O, the set of Turing machines that can make queries to O as a basic step satisfies these properties as well, and hence the results of computability theory, such as undecidability of the Halting problem etc.., hold for such machines as well— that is, they *relativize* for every oracle. This means that most computability results satisfy a nice 0/1 law— either they hold for all oracles or they hold for none of them.

When people started studying complexity theory, my understanding is that they expected that similar techniques may end up resolving complexity questions such P vs NP, which can be thought of as quantative analogs of questions such as R vs RE. Thus the result of Baker, Gill and Solovay, showing that there is a relativized world in which $P = NP$ and one in which $P \neq NP$, came (as far as I can tell) as a serious shock. Not too long after it, people realized that if we pick the oracle at random, then a complexity class relation can hold with either probability 0 or probability 1, but not with probability, say 2/3. Thus one can define a "random oracle analog" of every complexity class and study the relations between them in this "random oracle world". Moreover, this random oracle world is not completely unrelated to the real world— people have shown random oracle variants of several statements such as $P \neq NP$ that we believe hold in the "real" (non relativized) world. Bennet and Gill made the bold conjecture, known as the *random oracle hypothesis*, that these two worlds coincide— namely that complexity relations that hold with probability 1 for a random oracle are also true without an oracle. Arguably, there weren't strong arguments to support this conjecture (and indeed, as far as I know, it was never widely believed), and in fact there are several counterexamples to it, with perhaps the most convincing one coming from the work of Chang at al showing that relations between classes involving proof systems, such as $IP = PSPACE$, hold with probability 0 for a random oracle.

So, why care about random oracle? First of all, the complexity for random oracle is at least consistent (due to do this 0/1 law) and while it does not correspond to the real world, intuitions from the real world can be relevant there, and there are some results such as Book's formally connecting the two. In particular, it is interesting to note that the new result of Rossman et al is completely consistent with the intuition arising from thinking about random oracles, while it apparently was surprising to people thinking about influences of Boolean functions. Arguably, if Benjamini, Kalai and Schramm were more aware of this connection between their question and the polynomial hierarchy under random oracles, they could have realized their conjecture is likely to be false. So, studying complexity relations under random oracles can give intuitions and results useful in other areas.

Personally, I am interested in random oracles as they are related to my obsession with *structured vs unstructured* problems in computational complexity. A random function is the quintessential unstructured problem, while a problem such as integer factoring has strong algebraic structure. In particular, I am very interested in the question of whether $NP \cap coNP \neq P$ under a random oracle. Like Book's result, one can show that this is true assuming (as is widely believed) that $NP \cap coNP \nsubseteq BPP$ in the real (unrelativized) world. However, *proving* this is a different matter and seems to require a way of transforming an unstructured hard function to a hard function with the $NP \cap coNP$ structure. This can be extremely useful, and is (in my eyes) morally related to the question of whether one can construct a *public key encryption scheme* based on any *private key encryption scheme.* Indeed, random functions trivially give a private key encryption scheme, but most (if not all) of the widely studied problems that form the basis for public key encryption schemes lie in $NP \cap coNP$ (see my paper with Applebaum and Wigderson for more discussion). Unfortunately, the techniques of Rossman et al, as well as the other works in this area, seem unlikely to help resolve this issue, with the basic problem being that all these results prove something about constant-depth circuits and decision trees, and, alas, it actually does hold that $NP \cap coNP = P$ in the world of decision tree complexity (see more below).

1 On Oracles and circuits

We now start with expanding the connection between oracle and circuit results— we start with a proof of the following theorem, that you have already seen in this course:

Theorem 2. *There is an oracle O such that $P^O \neq NP^O$.*

The proof relies on the fact that we can set up some oracle $O : \{0,1\}^n \to \{0,1\}$ such that no algorithm that makes $\text{poly}(n)$ queries to O will be able to figure out whether there exists x such that $O(x) = 1$. Another way to phrase this statement is the following:

Theorem 3. *There is no decision tree of depth* $\text{polylog}(N)$ *(in fact $N-1$) that can compute the OR function $X_1 \vee \cdots \vee X_N$.*

The general "dictionary" transforming oracle results and circuit lower bounds is as follows:

Relativized world	circuit world
Oracle $O : \{0,1\}^n \to \{0,1\}$	Input $x \in \{0,1\}^N$
Random oracle O	random input O
$\text{poly}(n)$ time algorithm	$\text{polylog}(N)$-depth ($\text{quasipoly}(N)$ size) decision tree
NP-algorithm	$\text{polylog}(N)$-width ($\text{quasipoly}(N)$ size) DNF
$coNP$-algorithm	$\text{polylog}(N)$-width ($\text{quasipoly}(N)$ size) CNF
PH-algorithm	$\text{quasipoly}(N)$-size $O(1)$-depth circuit

The result that $P^O \neq NP^O$ for a random oracle follows from the following theorem:

Theorem 4. *For every* $\text{polylog}(n)$*-depth decision tree T,*

$$\Pr_{x \in \{0,1\}^n}[T(x) = F(x)] \leq 1/2 + o(1)$$

where $F(x)$ is the DNF that is the OR of N ℓ-AND's, set up so that $(1-2^{-\ell})^N = 1/2 \pm o(1)$.

Exercise 1: Conclude from the above theorem that with probability at least 0.99 over the choice of a random O, it holds that $NP^O \neq P^O$. See footnote for hint[1]

We know that $P \subseteq NP$ and $P \subseteq coNP$. In the circuit world this is the following theorem:

Theorem 5. *If T is a polylog(N)-depth decision tree, then there exists a* $\text{polylog}(N)$*-width DNF F, as well as a* $\text{polylog}(N)$*-width CNF F', computing the same function as T.*

Interestingly, in this world $P = NP \cap coNP$:

Exercise 2: If F is a function that is computable by both a $\text{polylog}(N)$-width DNF and a $\text{polylog}(N)$-width CNF, then it is also computable by a $\text{polylog}(N)$-depth decision tree.

[1]You can use the 0/1 law and conclude that otherwise, with probability at least $1 - o(1)$, for a random oracle O there would exists a poly-time algorithm $P(O)$ solving SAT with an oracle to O. For n large enough, that algorithm will be one of the first $f(n)$ Turing machines for some arbitrarily slowly growing to infinity function f. You can use that to get a decision tree succeeding in computing the OR function with good probability.

2 The Håstad switching lemma approach to lower bounds

The Håstad switching lemma is a powerful result that says something along the following lines:

Theorem 6 (Switching lemma, informal)**.** *If $C : \{0,1\}^N \to \{0,1\}$ is a poly-size (or even sub-exp) $O(1)$-depth circuit, then for some appropriately chosen parameter $\mu > 0$, if we fix at random $(1-\mu)$ fraction of the inputs to random values, then the new function $C' : \{0,1\}^{\mu N} \to \{0,1\}$ can be computed by a* polylog(N)*-depth decision tree.*

Suppose that the original circuit C had d layers alternating between AND and OR and say the bottom most was an OR. The proof of the theorem is obtained by repeating these random restrictions for d steps. So, in the first step we change the bottom most layer from a DNF into a decision tree, which in turn implies it can also be thought of as a CNF and hence merged with the second layer from the bottom. So, if in each step we keep alive a μ fraction of the variables, after keeping alive about μ^d fraction we end up with a decision tree.

It is very easy to prove that a parity on k variables cannot be computed by a decision tree of depth less than k (for every branch in which the decision tree outputs 0, one would be able to find an input that agrees with this branch but whose parity is 1) and hence we get the following corollary:

Corollary 7. *Parity has no constant depth circuits of subexponential size.*

In fact, its easy to show that a small depth decision tree cannot even *approximate* parity and hence we get that neither can a small depth circuits, which (using $P^{\#P} \subseteq PSPACE$) can also be phrased as follows

Corollary 8. *Relative to a random oracle O $PSPACE^O \not\subseteq PH^O$*

The problem with applying this result to separating, say, depth $d+1$ (or even depth $100d$) circuits from depth d, is that random restrictions act like a "slegehammer" that reduce any constant depth function into a small depth decision tree. So, if we replace parity with some particular function F that is computable with constant depth, then random restrictions will make it into a small depth decision tree as well, and hence not allow us to get a contradiction. What we need is to identify a specific function F that is computable in depth $100d$, and define a restriction that is more more delicate than a sledgehammer in the sense that it has the following properties:

- For every depth d circuit C, applying the restriction "peels off" at least one layer of C and hence reduces it to depth $d-1$.
- For the particular function F we are considering, the restriction does not "peel off" more than one layer (or 100 layers, if we're shooting for a $100d$ vs d separation) of the particular function F.

In fact Håstad managed to design such a restriction by creating very careful correlations between the restricted variables, and hence in particular showing that there are some functions computable by depth d circuits but not depth $d-1$ (and that the polynomial hierarchy is infinite with respect to some oracle). However, these correlations meant that the result did not hold for the average case / for a random oracle.

3 The RST approach

[ADD FIGURES HERE]

The specific function RST consider, which they call a "Sipser Function" is (essentially) an AND/OR tree of arity w. For every $p \in (0,1)$, we denote by the p-biased distribution the distribution on $\{0,1\}$ such that 1 is output with probability p. We let p be such that $(1-p)^w = p$. That is, an arity-w OR of w independent p-biased inputs yields a $1-p$ biased input, and similarly an arity-w AND maps the $1-p$ biased distribution into the p biased distribution. Note that since we have $p \sim e^{-pw}$ we get that $\ln(1/p) \sim wp$ or $p = \tilde{\Theta}(1/w)$. If we use depth ℓ tree on N inputs, then $w \sim N^{1/\ell}$.

So, if the input to the bottomost layer of the RST function is selected to be p-biased, then every layer alternates between the p-biased and the $1-p$-biased distribution. At the topmost layer RST add an AND or OR gate with an appropriate arity to map the distribution into the uniform (i.e., 1/2 biased) distribution. Also in the paper they add an additional bottomost layer with AND gates of arity $\log(1/p)$ to map the uniform distribution into the p-biased one, but this is not so important, and we can just assume that the input is selected according to the p-biased distribution.

Let F be the RST function of depth $100d$ and let C be the purported depth d circuit that approximates it.

Lets assume the bottommost layer of the RST function consists of width w OR gates. The first step in their restriction is to fix $1-p$ fraction of the inputs to 0. (Lets assume that things are nice and so every OR gates will have exactly pw unfixed inputs; in fact what we'll have to do is to condition on no OR gate having the all-zeroes input.) This has the benefit of not "killing off" any of the OR gates, but presumably will kill every AND gate in the bottomost layer of C that contains many un-negated variables, or every OR gate that contains many negated variables. However, this does not guarantee that we can "peel off" a layer of C. Indeed, we cannot rule out the case that C's bottom layer also had only OR's of un-negated variables, that cannot be killed by setting variables to 0.

Intuitively, as a next step we would like to randomly set some of those p fraction of unfixed variables to 1. However, if we do so randomly then most likely *every* OR of the RST function would contain a variable set to 1 and hence we would simply fix all those OR gates to 1, hence making it into a constant function, rather than "peeling off" only a single layer.

So instead, following Håstad, we are going to create significant correlations between the variables, and specifically ensure that we either fix all unfixed variables in a block to 1 or none of them. More formally, we will identify all the unfixed variables in the $j-th$ block of the RST bottom layer with a single variable y_j. Since the rest of the variables are zeroes, this means that the output of the OR gate is exactly y_j and hence we have now exactly peeled off the bottom layer of RST and have a function with depth $100d-1$ on the y_j variables. Now note that the y_j variables are supposed to be $1-p$ biased. So, now we can continue as before and pick $1-p$ fraction of them and set them to 1, leaving a p fraction unfixed. (Again, we will condition on no AND gate in the second layer getting killed by the all 1's input.) Since we have now fixed a good fraction of the variables to 1 that should take care of those gates that survived our first restriction in C. That is, globally we have fixed about $(1-p)^2$ fraction of the variables, where in the first stage we fixed $1-p$ fraction of them to 0's and in the second stage we fixed $1-p$ fraction of them to 1's. Intuitively, that should be enough to use the switching lemma arguments and to argue that we can "peel off" a layer of C. (There are some correlations involved, but as Håstad showed in his worst-case separation, since the correlations are designed to maximally help the RST function, and even there they peel off a layer, they should not hurt us in any other circuit.)

At the end of the day, because at each step we are performing the restriction according to the right distribution (p biased or $1-p$ biased) the end result respects the underlying input distribution and so we get an average case hardness. Thus, we would expect to get the following lemma:

Lemma 9. *Let F be the RST function (i.e., an AND/OR tree where all but the topmost levels have arity w and the bottom-most lavel is an OR) and let C be a depth-d quasipoly-size circuit, then in expectation after two restriction steps*

$$\Pr[F(X) = C(X)] = Pr[\hat{F}(Y) = \hat{C}(Y)]$$

where X denotes the p-biased distribution on $\{0,1\}^N$ and Y denotes the p-biased distribution on $\{0,1\}^{N/w^2}$, $\hat{F}$ denotes the restricted version of F and $\hat{C}$ denotes the restricted version of C. Moreover, with high probability $\hat{C}$ can be computed by a depth $d-1$ quasipoly-size circuit.

Actually, there are some additional complications and for technical reasons it turns out that RST are only able to prove that with this approach one can peel off a layer after *three* consecutive fixings rather than two, hence getting a $3d$ vs d separation. Also, the actual restriction they use in the paper is more complicated and manages to get a one to one ratio and so separate depth $d+1$ from depth d— one of the changes is that they fix only $1-q$ fraction of the inputs in each iteration for $q = \sqrt{p}$, and they also have to worry about the cases where the number of unfixed variables in a block deviates significantly from qw, as well slightly tune the parameters as the depth progresses, and hence the exact fixing done in advanced stages depends on the results of prior restrictions.

I am of course skipping many details, and in particular the proof of the switching lemma. To follow that proof, it is best if you first familiarize yourself with Razborov's proof of the standard switching lemma (e.g., see our book), as the RST proof is a variant on that proof. The proof is overall quite technical, and it is a good question of whether it can be simplified further:

Exercise 3: Give a simpler proof that there is a function computable in depth $f(d)$ poly-size circuit for some polynomial f (or even any function f) that cannot be approximated by a depth d circuit. Ideally, the proof should use the standard switching lemma (or at least Håstad's variant for showing worst case depth separation) as a black box.