Inside the Simons Institute

Dominik Scheder

October 9, 2013

Let f be a boolean function. Define the *circuit complexity* of ...No, wait! Let's back up a little bit. Let me tell you how we ended up here. We are in Berkeley, California, in a circular-shaped building on the university campus. It is week five in the life of the *Simons Institute for the Theory of Computation*. This institute is named after James Harris Simons, an American mathematician who started a second career as a hedge fund manager, and, once this had borne sufficient fruit, a third career as a philanthropist. His Simons Foundation endowed the institute with a grant of $60 million.

The goal of the institute is to promote research on the foundations of computer science and on other scientific areas where the paradigm of computation promises to be fruitful, for example biology or economics. Right now the institute houses its first two projects: *Theoretical Foundations of Big Data Analysis* and *Real Analysis in Computer Science*.

When I arrived a couple of weeks ago, the construction workers were still there, giving the final touch to the building. This did not prevent the program from hitting the ground running, with a one-week workshop on *Real Analysis in Testing, Learning and Inapproximability*. Week three saw a *Real Analysis Boot camp*, in which Johan Håstad, Adam Klivans, and Krzysztof Oleszkiewicz made all participants familiar with the methods and problems of the field. In contrast to usual boot camps, Johan Håstad did not shout at the recruits and order them to do push-ups, but rewarded correct answers and good questions with excellent Swiss chocolate (he changed to healthier apples towards the end).

Everything at the institute is designed to encourage people to collaborate and not toil in isolation: There is lots of common space, with whiteboards and sofas. Pleasant weather and blackboards encourage working outside. During workshops, coffee breaks are frequent and long enough to go beyond chit-chat and talk research—with a cup and a cookie in your hands. And if you wonder whether the research problem that just came up has already been solved, no need to search the web or the library: Just ask the expert in the office next door.

At this point, the reader might wonder what the scientific content of *Real Analysis in Computer Science* is. Let me briefly summarize three of the many talks the institute has seen in its first five weeks. Let us start with week one, when the institute held its first workshop, and let f be a boolean function.

Li-Yang Tan at the Week One Workshop

We are given a boolean function $f : \{0,1\}^n \to \{0,1\}$. Can we write f as a DNF formula? Surely we can. How many terms might we need? Never more than 2^n terms, and a minute or two of thought shows that 2^{n-1} are always enough. Neither is it too difficult to see that the Parity function $f(x_1, \ldots, x_n) = x_1 \oplus \cdots \oplus x_n$ requires a DNF formula of size 2^{n-1}. So this bound is exactly tight. Very nice! For once in circuit complexity, we know the answer exactly.

What about approximating f? Suppose we want to find a DNF formula D such that f and D agree on all but $\epsilon 2^n$ inputs. We say D is an ϵ-approximation of f. Can we make D smaller than 2^{n-1}? Sure! By converting f into a DNF formula D and deleting some terms, we can ϵ-approximate f with a DNF formula of size $(1-\epsilon)2^{n-1}$. The surprising result of Li-Yang Tan and Eric Blais (reference) is that one can do *much* better:

Theorem 1 (Blais and Tan [2])**.** *Every boolean function $f : \{0,1\}^n \to \{0,1\}$ can be ϵ-approximated by a DNF of size $O(2^n/\log n)$. The $O(\cdot)$ hides a constant factor which depends on ϵ.*

Besides its number of terms, a very natural complexity measure of a DNF formula is its *width*, that is, the maximum size of its terms. Suppose we ϵ-approximate f by a DNF formula D with s terms, where $s \in O(2^n/\log n)$ terms. Again, it is not too difficult to see that one can delete all terms of width larger than $\log_2(s/\epsilon)$ and still get a 2ϵ-approximation. In this case, the width of the resulting DNF formula is at most $n - \log\log n - \log\epsilon$. This is smaller than n, but can we do better?

Theorem 2 (Blais and Tan [2])**.** *Every boolean function can be ϵ-approximated by a DNF formula of width at most $(1-c_\epsilon)n$, where $c_\epsilon > 0$ and depends only on ϵ.*

To see how surprising this theorem is, think of choosing f randomly. That is, for every $x \in \{0,1\}^n$, toss a coin to decide whether $f(x)$ should be 0 or 1. What is a term of a DNF formula? It is a conjunction of literals, and its satisfying assignments correspond to a subcube of $\{0,1\}^n$. If the term has width $(1-c)n$, the subcube contains 2^{cn} points. Since $c > 0$ is a constant, it contains an exponential number of points. Thus, the random function f will be very balanced on every subcube. The theorem says that we can still select subcubes that intersect in a very special way, such that f is very unbalanced on the union of these subcubes: Almost always 1 on that union, and almost always 0 outside.

Aviad Rubinstein in the Thursday Seminar

Every Thursday morning, we meet for a seminar of one long and four short talks. Short really means short: The speaker has eight minutes to get the audience excited about his or her result. I'll try to get the reader excited about the result by Aviad Rubinstein I heard this morning.

Let $X_1, \ldots, X_n$ be unbiased coins, i.e., independent variables taking on values in $\{-1, 1\}$ with equal probability. Let $Z = a_0 + a_1X_1 + \cdots + a_nX_n$ for some real numbers $a_0, a_1, \ldots, a_n$. When does Z look like a Boolean function $f : \{-1, 1\}^n \to \{-1, 1\}$? For instance, let $Z = \frac{X_1+\cdots+X_n}{\sqrt{n}}$. Note that Z falls into the interval $\left[-\frac{1}{2}, \frac{1}{2}\right]$ with constant probability, and thus is not very close to any boolean function f. In a second example, let $Z = \frac{X_1+\cdots+X_n}{n} + X_{n+1}$. Most of the time the first summand is really small, and thus Z is usually close to X_{n+1}, which is itself a boolean function. The FKN Theorem (Friedgut, Kalai, and Naor [3]) states that if Z is close to a boolean function, it is close to some X_i:

Theorem 3 ([3]). *Let $Z = a_0 + a_1X_1 + \cdots + a_nX_n$. If $f : \{-1, 1\}^n \to \{-1, 1\}$ satisfies*

$$\mathbb{E}\left[(Z - f(X_1, \ldots, X_n))^2\right] \leq \epsilon ,$$

then there exists some $1 \leq i \leq n$ such that $\mathbb{E}\left[(Z - X_i)^2\right] \leq \epsilon$.

Aviad Rubinstein and Muli Safra [7] proved a generalization of this theorem: The X_i need not be unbiased coins. In fact, they need not be coins. All that is required is that they be "well-behaved" independent random variables. For example, each X_i could be itself a boolean function f_i, over a set of boolean variables that is disjoint from that of the other functions f_j. A similar result was discovered independently by Jendrej, Oleszkiewicz, and Wojtaszczyk [5].

Johan Håstad at the Boot Camp on Real Analysis in Computer Science

In the first four lectures of his mini-course Johan Håstad explained his classical result that Max-3-SAT is NP-hard to approximate within a factor of $\frac{7}{8} + \epsilon$ [4]. On top of giving this beautiful proof in full detail, he treated the audience with insider stories on how he came up with this proof, which obstacles he faced and how he overcame them.

In the fifth lecture he presented a very recent result about a different notion of approximation. Suppose F is a CNF formula, and we are promised that there exists a "super-assignment" that in every clause satisfies at least half of all literals. The goal is to find any satisfying assignment. A simple random walk algorithm by Papadimitriou [6] solves this in expected polynomial time. What if we relax the promise marginally and are only promised that the super-assignment satisfies 0.49 of the literals in every clause. On such instances, Papadimitriou's algorithm might have exponential running time. Is this problem "Super-SAT" hard in general?

The answer is yes, it is hard, as shown recently by Per Austrin, Venkatesan Guruswami, and Johan Håstad [1]. This result was the subject of Johan Håstad's fifth lecture: Given a CNF formula in which every clause consists of exactly $2k+1$ literals, and given the promise that some assignment satisfies at least k literals per clause, it is still NP-hard to find a satisfying assignment.

Their proof uses some established machinery in an unusual way. They start with an instance of the Label Cover problem—just as most inapproximability proofs, including Håstad's hardness proof for approximating Max-3-SAT. Label Cover is a certain coloring problem on bipartite graphs, which is known to be extremely hard to approximate. Austrin, Guruswami, and Håstad use Long Codes to encode the colors $\{1, \ldots, r\}$ of the Label Cover problem into binary. So far this follows well-established paths. However, their construction of a CNF gadget from the long codes is very different from the Max-3-SAT case. After all, the goal of the Super-SAT problem is not to maximize a sum (as in the case of Max-3-SAT, where we want to maximize the sum of the weight of satisfied clauses), but to minimize a maximum: We want to minimize the maximum number of unsatisfied literals a clause contains. In fact, that maximum should be 0. This instance of "bottleneck optimization" requires quite different an analysis. In fact, their reduction gadget can be analyzed combinatorially without Fourier analytic tools.

References

[1] Per Austrin, Venkatesan Guruswami, and Johan Håstad. work in progress.

[2] Eric Blais and Li-Yang Tan. Approximating boolean functions with depth-2 circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:51, 2013.

[3] Ehud Friedgut, Gil Kalai, and Assaf Naor. Boolean functions whose fourier transform is concentrated on the first two levels. *Adv. in Appl. Math*, 29, 2002.

[4] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[5] Jacek Jendrej, Krzysztof Oleszkiewicz, and Jakub O. Wojtaszczyk. submitted.

[6] Christos H. Papadimitriou. On selecting a satisfying truth assignment (extended abstract). In *Proceedings of the 32nd annual symposium on Foundations of computer science*, SFCS '91, pages 163–169, Washington, DC, USA, 1991. IEEE Computer Society.

[7] Aviad Rubinstein. Boolean functions whose fourier transform is concentrated on pair-wise disjoint subsets of the inputs. Master's thesis, Tel-Aviv University, 2012.