Tools and Sensitivity

Cutting right through a 30-year-old conjecture

Hao Huang is a mathematician and computer scientist at Emory University. Last week he released a paper of only six pages that solves the Boolean Sensitivity Conjecture, which goes back at least to a 1992 paper by Noam Nisan and Mario Szegedy.

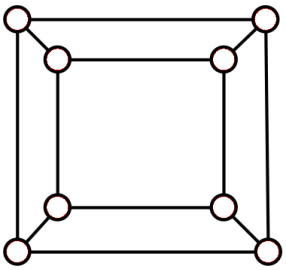
Today we discuss his brilliant proof and what it means for sensitivity of the *tools* one employs.

Several of our blogging friends have covered this news in posts already, and Ryan O'Donnell even summarized the proof in one tweet. Scott Aaronson's thread includes a comment by Huang on how he came by his proof.

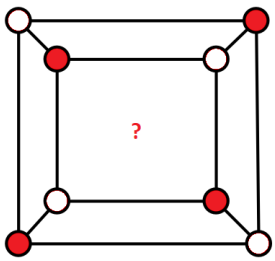
We will try to draw implications for the related matter of how *you* might come by proofs of *other* conjectures. We have previously discussed the possibility of "overlooking short solutions to major problems." Here we will discuss how to *find* them.

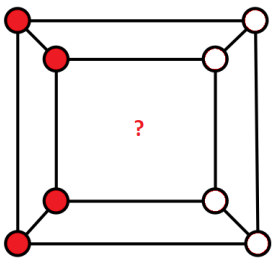
A Graph Puzzle

To get a flavor of what Huang proved, consider the graph of an ordinary cube:



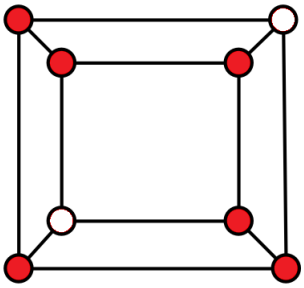
The question is, *can you color 5 vertices red so that no red node has 3 red neighbors?* Your first impulse might be to color 4 nodes red according to parity so that none has a red neighbor, per below left:



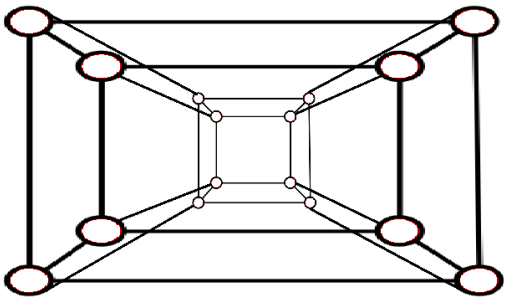


But then any 5th node will have 3 red neighbors. Another "greedy" idea is to pack a subgraph of the allowed degree 2 into half the cube, as at right. Any 5th node will again create a degree-3 vertex in the subgraph induced by the red nodes.

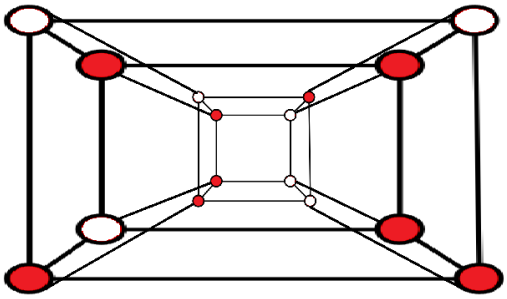
The answer is that actually one can pack 6 nodes that induce a simple cycle:



Now let's up the dimension by one—that is, take $n = 4$ and $N = 2^n = 16$. How many nodes can we color red and keep the induced degree 2?



Again the parity trick gives us degree 0 with 8 nodes, but then we can't add a 9th. We can greedily try to pack the outer cube with our 6-node solution, but then—perhaps surprisingly—we can add only 2 more red nodes from the inner cube. So we can only do 5 from the outer cube. We can get 9 overall by:



The fact that one red node is isolated seems to give room to improve, but there is no way to make 10.

The Theorem

The calculations have left an interesting jump from degree 0 with eight red nodes and degree 2 with nine. How about degree 1? Can we do that with 9 nodes? We can pack four disjoint edges but then there is nowhere to stick an isolated node.

So for 9 nodes, which is $\frac{N}{2}+1$, the best we can do is degree 2, which is $\sqrt{n}$. This is what Huang proved:

Theorem 0.1. *Every subgraph induced by $\frac{N}{2}+1$ nodes of the n-dimensional hypercube graph has a node of degree at least $\sqrt{n}$.*

This is completely tight. When n is a perfect square there is a way to achieve $\sqrt{n}$ as the maximum degree (shown here). Otherwise the least integer above $\sqrt{n}$ is best. Thus every subgraph of the 5-cube induced by 17 nodes has a node with three neighbors, but you can go as high as 257 nodes in the 9-cube while keeping the maximum degree to 3.

Now we need to discuss, *why is this notable?*, and *why was it hard to see?*

Boolean Functions and Sensitivity

The parity function $f_\oplus$ is extremely sensitive: if you change one bit of any argument x you change the value of $f_\oplus(x)$. Define x^i to mean x with bit i flipped. The OR function $f_\vee$ is intuitively less sensitive, but it too has an argument x such that $f_\vee(x^i) \neq f_\vee(x)$ for all i, namely $x = 0^n$. For any Boolean function $f : \{0,1\}^n \to \{0,1\}$ define its *sensitivity* (at n) by

$$s(f) = s_n(f) = \max_{x \in \{0,1\}^n} ||\{i : f(x^i) \neq f(x)\}||.$$

The OR-of-AND function f_2 is less sensitive. Say it is an OR of m blocks, each of k variables, and the blocks use disjoint variables so $n = km$. If $f_2(x) = 1$, then some block is all 1, so flipping any other bit makes no difference, and the most sensitive we can get is k. If $f_2(x) = 0$ then the most sensitive case is when all blocks have exactly one 0. So $s(f_2) = \max\{k, m\}$. When $k = m = \sqrt{n}$, $s_n(f) = \sqrt{n}$.

If we make each block of f_2 return *true* if exactly one bit is 1, then we again get sensitivity n on the all-0 assignment. Now, however, consider the related function f_2' (with k even, $k = 2\ell$) that is still an OR over blocks, but each block is true when some consecutive pair $x_{2\ell-1}, x_{2\ell}$ are both 1 with all other pairs being both 0. Then we can't do the same trick with the all-0

assignment changing just one bit. So when $n = 4\ell^2$ and $m = k = 2\ell$ we again get sensitivity (no more than) $\sqrt{n}$.

We can, however, consider f_2' to be more sensitive if we can flip more than one bit at a time. Partition $[n]$ into *blocks* B of two consecutive bit-places each, and given any x, define x^B to be the result of flipping the bits in B. We get $n/2$ blocks, and the all-0 assignment becomes a *true* case of f_2' if any block is flipped. Generally define the *block sensitivity* by considering any partitions $\mathcal{B} = \{B_j\}$ of $[n]$ into disjoint subsets and writing

$$bs(f) = \max_{x,\mathcal{B}} ||\{j : f(x^{B_j}) \neq f(x)\}||.$$

Note that not every member of the partition has to flip the function—we can discard the ones that don't flip and count only the disjoint subsets that do flip the value. So back to our example, we have

$$bs(f_2') = \frac{n}{2} = \frac{1}{2}s(f_2')^2.$$

Andris Ambainis and Xiaoming Sun improved the constant from $\frac{1}{2}$ asymptotically to $\frac{2}{3}$, but their relation is still quadratic.

The Connections

This example of quadratic discrepancy is still the best known lower bound on $bs(f)$ in terms of $s(f)$. But no one had proved anything better than an exponential upper bound until Huang's result, from which it follows that:

Theorem 0.2. *For all Boolean functions f, $bs(f) \leq 2s(f)^4$.*

This bound is concrete, not just asymptotic. It still leaves a gap between quadratic and quartic. It is, however, the combination of two quadratic upper bounds. One was shown by Nisan and Szegedy in their paper:

$$bs(f) \leq 2\deg(f)^2,$$

where $\deg(f)$ means the degree of the unique multi-linear real polynomial that agrees with f on the cube $\{-1, 1\}^n$ with -1 for *true*. The other is the conjecture

$$\deg(f) \leq s(f)^2$$

in a 1992 paper by Craig Gotsman and Nati Linial, which is exactly what Huang proves.

How do we get to this from Theorem 0.1? The connection to graphs was also shown by Gotsman and Linial. With reference to our node colorings above, let G be the graph induced by the red nodes and let $g(x) = 1$ if node x is red, $g(x) = 0$ otherwise. Now if the maximum degree $d(G)$ of a node in G is

small then every red node has many white neighbors, so g is very sensitive. However, going to each neighbor in the hypercube flips the parity. Hence the function

$$g'(x) = g(x) \oplus f_{\oplus}(x)$$

is **not** very sensitive. Moreover, it has the same sensitivity as the function $h'(x) = h(x) \oplus f_{\oplus}(x)$ where $h(x)$ is true on the white nodes. This nice duality between G and the graph H induced by the white nodes enables us to fix "G" to mean whichever of the two has more nodes in the following theorem statement:

Theorem 0.3. *Provided $m > 2^{n-1}$, every graph G induced by m nodes of the n-cube has $d(G) \geq \sqrt{n}$ if and only if every Boolean function f has* $\deg(f) \leq s(f)^2$.

The proof uses some Fourier analysis with $f = g'$ as above. It, too, takes only one page of a really short paper, and we could go into how Fourier analysis is a highly sensitive tool in its own right. But we'll move on to Huang and matrix tricks.

The Proof

From my undergrad days I've kept an interest in spectral graph theory. One of the basic facts is that the degree $d(G)$ of a graph G is always at least as great as the largest eigenvalue λ of its adjacency matrix A_G. For a d-regular graph they are equal. Huang's first trick is to note that the classic proof of this also allows -1 values on edges:

Lemma 0.4. *Let A be a symmetric matrix obtained from A_G by multiplying some entries by -1. Then $d(G) \geq \lambda$.*

Proof. Choose an eigenvector v such that $Av = \lambda v$ and take an index i that maximizes $|v_i|$. Then

$$|\lambda v_i| = |(Av)_i| = |\sum_j A_{i,j} v_j| \leq |\sum_j A_{i,j}| \cdot |v_i| \leq \sum_{(i,j) \in E(G)} |A_{i,j}| \cdot |v_i| \leq d(G)|v_i|.$$

Dividing out $|v_i|$ gives the lemma. □

So now what we want to do is find conditions that force $\lambda = \sqrt{n}$ when G is a m-vertex subgraph of the n-cube with $m \geq \frac{N}{2} + 1$, where $N = 2^n$. The trick that Huang realized is that he could do this by making A sit inside a matrix A_N with at least $\frac{N}{2}$ eigenvalues of $\sqrt{n}$.

To see how, form A_{N-1} by knocking out the last row and column of A_N. Since A_N and A_{N-1} are both real and symmetric, their eigenvalues are real,

so we can order them $\lambda_1, \ldots, \lambda_N$ and $\mu_1, \ldots, \mu_{N-1}$ in nonincreasing order. The basic fact is that they always *interlace*:

$$\lambda_1 \geq \mu_1 \geq \lambda_2 \geq \mu_2 \geq \lambda_3 \geq \cdots \geq \mu_{N-1} \geq \lambda_N.$$

See this for a one-page proof. The neat point is that you can repeat this: if you get A'' by knocking out another row and corresponding column, and $[\nu_i]$ are its eigenvalues in order, then

$$\mu_1 \geq \nu_1 \geq \mu_2 \geq \nu_2 \geq \mu_3 \cdots .$$

It follows that $\lambda_1 \geq \nu_1 \geq \lambda_3$. If you do this again, you get a matrix whose leading eigenvalue is still at least as big as λ_4. Do it $\frac{N}{2} - 1$ times inside A_N, and you're still above $\lambda_{N/2}(A_N)$, which we just said we will arrange to be $+\sqrt{n}$. Thus if we knock out the $\frac{N}{2} - 1$ white nodes, we will get the graph on the red nodes with adjacency matrix A_m and conclude:

$$\lambda_1(A_N) \geq \lambda_1(A_m) \geq \lambda_{N/2}(A_N).$$

Plugging into the lemma gives:

$$d(G) \geq \lambda_i(A_m) \geq \lambda_{N/2}(A_N) = \sqrt{n}.$$

(In fact, as also noted on Scott's blog, this case of interlacing can be inferred from simpler reasoning—but our point is that the interlacing theorem was in Huang's bag of tricks.)

Building the Matrix

Finally, how do we lay hands on A_N? We want a matrix of trace zero such that $A_N^2 = nI$. Then all its eigenvalues are $+\sqrt{n}$ and $-\sqrt{n}$—and in equal numbers because they sum to the trace which is zero. So we will have $N/2$ eigenvalues of $\sqrt{n}$, as needed. And we would want A_N to be the matrix of the n-cube but that doesn't work: each i, j entry of its square counts all paths of length 2 from node i to node j and that number can be nonzero.

This is where the trick of putting -1 on edges comes in, and we can explain it in a way familiar from quantum. We arrange that every 4-cycle of the n-cube has exactly one edge with -1. Then the pairs of paths from one corner to the opposite corner will always *cancel*, leaving $A_{i,j}^2 = 0$ whenever $i \neq j$. And $A_{i,j}^2 = n$ because there are n ways to go out and come back along the same edge, always contributing $1 \cdot 1$ or $(-1) \cdot (-1) = 1$ either way. Huang defines the needed labeling explicitly by the recursion:

$$A_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \text{and for } n > 1, \quad A_n = \begin{bmatrix} A_{n-1} & I \\ I & -A_{n-1} \end{bmatrix}.$$

This puts a $-$ sign on exactly one-fourth of the entries in the needed way. That's it—that's the proof.

Why was it hard to spot? Dick and I believe it was the -1 trick. In the 1980s, I thought about ways to convert undirected graphs into directed ones by putting arrows on the edges, but not -1 signs. The chance of thinking of it maybe rises with knowing quantum. Now we can see, OK, A_1 is the quantum NOT gate and the recursion treats signs in similar fashion to the recursion defining Hadamard matrices. This all goes to our main point about having tools at one's command, the more tools the better.

Open Problems

The main open problem is whether the gap between quartic and quadratic can be closed. Huang notes that his spectral methods need not be confined to sub-matrices of the n-cube, and our thoughts of involving quantum are similar. Can quantum tools improve the results even further?

There is a much wider suite of Boolean complexity measures besides $s(f)$, $bs(f)$, and $\deg(f)$ discussed here. For example, consider how many bits of x you need to fix in order to preserve the value $f(x)$. That is, define $C(f)$ to be the maximum over x of the minimum size of a set $I \subseteq [n]$ such that whenever x' agrees with x on I, $f(x') = f(x)$. Clearly I needs to include at least one bit from each B_j. This proves $C(f) \geq bs(f)$. There are many other relations that might be improved—or at least better understood—by new methods.