Exploration of Boolean Function Sensitivity and the Sensitivity Conjecture

Darren Denq

June 2020

1 Introduction

In this paper we will go over the recent breakthrough solution to the Sensitivity Conjecture by Huang in his paper Induced subgraphs of hypercubes and a proof of the sensitivity conjecture [5]. We start with the following definition:

Definition 1.1 (Boolean Function)**.** A Boolean function f takes a fixed size vector of elements from a set of two distinct values and outputs an element from that same set; that is, for set $S = \{a, b\}$, $a \neq b$, $f : S^n \to S$. In most cases (and in this paper), S is either the set $\{0, 1\}$ representing the possible states of a computational bit or $\{-1, 1\}$.

With $S = \{0, 1\}$, it is easy to see that Boolean functions are fundamental in computer science - basic circuits can be expressed as Boolean functions in terms of their logic gates (eg OR, AND, XOR). A natural next question would then be: for such a Boolean function, what is the (minimum) number of these logic gates required to form a circuit that matches the function? This comes with a notion of "complexity" for Boolean functions, perhaps a loose classification based on the number of logic gates required. Indeed, it is questions of this flavor that make up the study of *circuit complexity*, a branch of *computational complexity theory* in computer science. Another question may be: given the probabilities that a value will occur for each element of the input, what is the probability of obtaining each value of the output? The link between this question and the complexity of Boolean functions will hopefully become clearer once we examine some complexity measures.

As it turns out, there are many different ways to gauge the relative complexity of a Boolean function for a variety of purposes, one of which is **sensitivity**. For an input x to some Boolean function, we will take the notation x^i to mean x with the i-th element from the left changed to the opposite value, zero-indexed (eg if x were the bit vector 011001, then x^0 would be 111001). Note that $0 \leq i <$ the length of x.

Definition 1.2 (Sensitivity)**.** The sensitivity of a Boolean function f with regards to an input x, denoted $s(f, x)$, is the number of different values of i such that $f(x) \neq f(x^i)$. The sensitivity of f, denoted $s(f)$, is

$$s(f) = \max_x s(f, x)$$

across all possible inputs x, the domain of f.

The sensitivity of a function can be thought of as the maximum number of bits that can be flipped and yield a different result.

Another measure of Boolean function complexity is **block sensitivity**. As an extension of our previous notation, letting n be the length of x, we say that x^B where block $B \subseteq \{0, 1, ..., n\}$ means x with the i-th elements flipped for $k \in B$ (eg if x were the bit vector 011001, then $x^{\{0,1,4\}}$ would be 101011).

Definition 1.3 (Block Sensitivity)**.** The block sensitivity of a Boolean function f with regards to an input x with length n, denoted $bs(f, x)$, is the number of disjoint sets $B \subseteq \{0, 1, ..., n\}$, $B \neq \emptyset$, such that $f(x) \neq f(x^B)$. The block sensitivity of f, denoted $bs(f)$, is

$$bs(f) = \max_x bs(f, x)$$

across all possible inputs x, the domain of f. The sets B are called blocks.

Remark. A block B is said to be *minimally sensitive* for x if for any $B' \subset B$, $f(x) = f(x^{B'})$. Then x^B is sensitive to all indexes in B and thus $|B| \leq s(f, x^B) \leq s(f)$.

Lemma 1.1. *For some f and x such that $bs(f, x) = bs(f)$, there exists some set β of minimally sensitive blocks such that for each $B \in \beta$, $f(x) \neq f(x^B)$ and $|\beta| = bs(f)$.*

Proof. We first start with some β that satisfies the properties above but does not have all (or any) its blocks minimally sensitive (the existence of such β is given from definition). For any non minimally sensitive block B there are two ways in which the property is not satisfied: either there is exactly one minimally sensitive subset of B for which f is sensitive on x, or there are at least two disjoint minimally sensitive subsets of B for which f is sensitive on x. In the former case we can simply replace B with the subset. In the latter case, if remove B from β and replace it with the subsets, we see that f is still sensitive to all blocks in β with regards to x but now $|\beta| > bs(f)$ which is a contradiction to our construction. □

The block sensitivity can be thought of as sensitivity but groups of switches can be flipped at once. We see the difference in the following example:

> Consider the Boolean function that takes 4-bit vectors and returns 1 if all the ones are left of all the zeros or if all the ones are left of all the zeros, and returns 0 otherwise. That is, the function returns 1 for 0000, 0011, 1111, 1110, 1000, etc. and returns 0 for 0110, 0100, 1010, 1001, etc. By writing out the combinations or some thought we see that the sensitivity of this function is 2 and the block sensitivity of this function is 3. This can be seen in that flipping either of the two middle bits of 0000 changes the output (0100, 0010), and so does flipping the bits at the end at the same time (1001).

Using the definitions, we can quickly establish some intuitive bounds for $s(f)$ and $bs(f)$:

Lemma 1.2. *For a Boolean function f that takes input of length n*

$$0 \leq s(f) \leq bs(f) \leq n$$

Proof. With the obvious lower bound of 0 (eg constant function) and a cap of n as a maximally sensitive function changes output with any bit flip (eg parity function), and the maximum number of disjoint subsets (not counting the empty set) of a set of n elements is n. To see that $s(f) \leq bs(f)$, we can take the same x used to find $s(f)$ and consider each i of $s(f, x)$ and consider them blocks of single bits. An example where $s(f) < bs(f)$ is given above (with a little work from the reader). □

Sensitivity and block sensitivity are two of many different measure of complexity of Boolean functions. In the following sections we will tie it in to block sensitivity by going over a proof of the Sensitivity Conjecture.

2 Sensitivity Conjecture

2.1 Some More Complexity Measures

There have been many other complexity measures discovered to be useful over the years. We will introduce a few more of them in hopes to build some motivation for the Sensitivity Conjecture. One of such is certificate complexity.

Definition 2.1 (1-Certificate and 0-Certificate)**.** For a Boolean function f, the 1-certificate is a fixed assignment of a subset of the elements of the input such that the output from any input that satisfies the certificate will be 1. The 0-certificate is the same except the desired output is 0. The number of elements fixed is said to be the size of the certificate, denoted by $|C|$ for certificate C.

Definition 2.2 (Certificate complexity)**.** The certificate complexity of a Boolean function f with regards to an input x, denoted $c(f, x)$, is the size of the smallest certificate for f that x matches. The certificate complexity of f, denoted $c(f)$, is

$$c(f) = \max_x c(f, x)$$

across all possible inputs x, the domain of f. If we wish to specify the boolean output of the certificate, say out of $\{0, 1\}$, we write $c_1(f)$ or $c_0(f)$ to represent the smallest 1-certificate or 0-certificate for f that x matches, respectively. $c(f) = \max(c_1(f), c_0(f))$.

From the definition, we can see that certificate complexity is useful in contexts such efficient guarantees on data; that is, the number of bits that need to be queried for a result to be correct. However the uses of certificate complexity are beyond the scope of exploration of this paper. A perhaps surprising result is that the certificate complexity can be bounded polynomially by the block sensitivity (and thus vice versa).

Theorem 2.1. *For a Boolean function f, the relationship between $bs(f)$ and $c(f)$ is:*

$$bs(f) \leq c(f) \leq bs(f)^2$$

Proof. We proceed in two parts.
$bs(f) \leq c(f)$: For input x any certificate must include at least 1 element from each of the sets B which make up $bs(f, x)$ (see definition 1.3). Otherwise, if the certificate does not include an element of B', then from the definition of block sensitivity those bits can be set to change the output, which violates the definition of certificate.
$c(f) \leq bs(f)^2$: Let x be an input such that $bs(f, x) = bs(f)$ and β be the set of minimally sensitive (see lemma 1.2) blocks B which make up $bs(f, x)$ ($|\beta| = bs(f)$). Let c be the certificate with fixed indices $\bigcup_\beta B$. We then show that c is indeed a certificate of f via contradiction. Suppose C is not a certificate of f; that is, there exists some y such that $f(y) \neq f(x)$ even though y satisfies c. Then there is a block B' such that $y = x^{B'}$. Because $f(y) \neq f(x)$, f is sensitive to B' with regards to the input x. But from the construction of y, we see that B' is disjoint from all $B \in \beta$ and thus $bs(f, x) = |\beta| + 1 > bs(f)$ which is a contradiction. Thus we have shown that c is a certificate of f as desired. Then we see that $c(f) \leq |C| = \bigcup_\beta B \leq bs(f)s(f)$. Applying lemma 1.2 we get $c(f) \leq bs(f)^2$ as desired. □

Yet another complexity measure of Boolean functions is decision tree complexity, which loosely speaking treats Boolean functions as state machines and classify them based on the resulting size.

Definition 2.3 (Boolean Decision Tree)**.** A Boolean decision tree for an input length of n is a binary tree where each node either has one or two child nodes and contains a *query index*, i where $0 \leq i < n$, or is an output node and has a Boolean value. For the nodes with a query index, the value of the i-th element is examined and the child chosen based on the value. If the number of child nodes is 1, then the result of the query does not matter. An input starts at the root of the tree, and nodes are traversed through successive queries until an output node is reached, the value of which is said to be the *decision*.

Thus we see that a Boolean decision tree T also maps a fixed size Boolean input to a Boolean output, exactly like a Boolean function. We say that T *computes* a Boolean function f if T has the same input and output mapping as f. We take for granted the ability to construct a Boolean decision tree equivalent to any f, and leave the process for doing so as an exercise to the reader.

Definition 2.4 (Decision Tree Complexity)**.** The decision tree complexity of a Boolean function f, denoted $D(f)$, is the *height* (ie the greatest number of nodes between an input and output node, inclusive) of the smallest height Boolean decision tree T such that T computes f. Note that T is not necessarily unique.

Decision trees are a useful expression for Boolean functions in various situations like where the (differing) probabilities of the results of each index query are given and the probabilities of certain states along the tree occurring are of interest. Again, exploration of such applications are beyond the scope of this paper but we hopefully gain some understanding of the importance and use cases of these different complexity measures.

As it turns out, decision tree complexity $D(f)$ is also bound by certificate complexity $c(f)$ and thus also by block sensitivity $bs(f)$.

Theorem 2.2 ([1]). *For a Boolean function f, the relationship between $D(f)$ and $c(f)$, $c_0(f)$, $c_1(f)$ is:*

$$c(f) \leq D(f) \leq c_0(f)c_1(f) \leq c(f)^2$$

Proof. We again proceed in two parts. Let T be some Boolean decision tree that computes f and has height $D(f)$.
$c(f) \leq D(f)$: The query results that take an input x from the root of T to an output state can be put together to form a certificate for that output (from the nature of binary trees). This includes partial matches; that is, output nodes that are less than $D(f) - 1$ nodes away from the root. Thus considering any x used to maximize in the calculation of $c(f)$ we see that the desired result holds.
$D(f) \leq c_0(f)c_1(f)$: We proceed by providing an algorithm for constructing a decision tree for f. When we say to *query a certificate* in a decision tree, we mean to add nodes to the tree one after another such that they test the fixed elements of the certificate. Let ξ_1 be the set of 1-certificates of length at most $c(f)$ and ξ_0 be the set 0-certificates of length at most $c(f)$. In the trivial case of a constant function, the tree has two nodes: the root and an output node; and we are done. We first aim to show the process for an input x. The algorithm is as follows: (1) Remove some c_1 from ξ_1 (temporarily). Query all the bits in c_1, adding missing nodes when applicable. Remove from ξ_1 all certificates that do not satisfy c_1, or are already completely covered by the past queries. Then recurse from (1) starting at the resulting node querying c_1. The current algorithm terminates when either x fulfills a certificate in which case an output node of 1 is added, ξ_1 is empty in which case an output node of 0 is added, or there are no certificates in ξ_0 that can possibly be satisfied given the constraints of previous queries. The key observation for termination of the algorithm is as follows: every 1-certificate and 0-certificate must intersect as otherwise there would exist some input that satisfies both certificates. This means that for every 1-certificate queried in succession the set of indices which fix 0-certificates increases by at least 1 as the c_1 selected every recursive step does not match any previous certificates. Thus after at most $c_0(f)$ recursions fulfillment of a 0-certificate will be impossible and all inputs that reach this point must have output 1. As x was arbitrary, this can be done for all x. Thus the resulting decision tree will compute f, as desired. Since each c_1 adds at most $c_1(f)$ height to the decision tree, the upper bound of the height of this decision tree is $c_0(f)c_1(f)$, bounded above by $c(f)^2$ and below by $D(f)$ from definition. □

This also implies that $D(f)$ and $bs(F)$ are polynomially bounded by each other from Theorem 2.1. Now we start to see an interesting pattern: it seems like these conceptions of complexity measures can all be bounded polynomially by each other. Indeed, across the work of many years it has been shown that other complexity measures (including two-sided bounded error randomized decision tree complexity, two-sided bounded error quantum decision tree complexity, polynomial degree, and approximating polynomial degree, CREW PRAM [2]) are also bounded polynomially by each other, $bs(f)$, $c(f)$, and $D(f)$. The observant reader will recall that we have not discussed the sensitivity complexity measure in this regard. Indeed, it is precisely the question of whether sensitivity is included in this set of mutually-polynomially-bounded complexity measures which was the subject of work for many years. This was particularly puzzling given the simple construction of sensitivity and close relation to block sensitivity.

2.2 The Sensitivity Conjecture

Proposed by Nisan and Szegdy in 1992 [9]:

Conjecture 2.1 (Sensitivity Conjecture)**.** *There exists an absolute constant $M > 0$, such that for every Boolean function f,*

$$bs(f) \leq s(f)^M$$

2.3 Some Earlier Work

To provide further context and further reading we briefly go over some earlier results. Previous work on the problem yielded exponential bounds at best, or solved a weaker version of the problem (namely with conditions placed on f).

- [11] Simon: $bs(f) \leq s(f)4^{s(f)}$
- [8] Nisan: For monotone f, $c(f) = s(f) = bs(f)$
- [7] Kenyon and Kutin: $bs(f) \leq \frac{e^{1+s(f)}\sqrt{s(f)}}{2\pi}$

3 The Proof

After standing for decades, in 2019 Hao Huang gave an elegant proof of the the Sensitivity Conjecture via it's analogous problem in graph theory. We examine the argument in this section.

3.1 Some Review and Machinery

We start with a review of basic terminology in graph theory.

Definition 3.1 (Graph)**.** A graph is defined by a collection of a set V of *vertices* and a set E of *edges* representing connections between the vertices; $E \subseteq V \times V$. In this sense a graph is the pair $G = (V, E)$. For a some vertex v, the edge (v, v) is said to be a *self-edge*. Some graphs do not allow self-edges, in which case the set of edges becomes $E \subseteq (V \times V) \setminus \{(v, v) : v \in V\}$.

Graphs can be thought of as webs of vertices with edges connecting them. With this understanding, we have some more definitions:

Definition 3.2 (Vertex Degree)**.** The degree of a vertex of a graph is the number of edges connected to it. That is, for some vertex v the degree is the number of edges $(p, q) \in E$ where $p = v$ or $q = v$.

Definition 3.3 (Graph Degree)**.** The degree of a graph G, denoted $\Delta(G)$, is the maximum of the degrees of its vertices.

Definition 3.4 (Induced Subgraph)**.** For some graph $G = (V, E)$ we say that the induced subgraph $G'(V', E')$ is a subset of G, $V' \subset V$ and $E' \subset E$, with edges only between vertices in V'.

Lemma 3.1. *For $G = (V, E)$ and two induced subgraphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ such that $V_1 \subset V_2$, we have that $\Delta(G_1) \leq \Delta(G_2)$.*

Proof. When adding a vertex and it's edges to G_1, one of three things can occur: the new vertex connects to at least one vertex of G_1 with degree $\Delta(G_1)$ in which case the degree of the new graph is $\Delta(G_1)+1$, the new vertex connects only to vertices of G_1 with degree less than $\Delta(G_1)$ in which case we quickly see that the degree of the new graph is $\Delta(G_1)$, or the new vertex does not connect to any vertices in G_1 in which case we quickly see that the degree of the new graph is $\Delta(G_1)$. Thus the degree of a subgraph only goes up as vertices/edges are added to it (ie degree of a subgraph is monotonically increasing with regards to nodes added). □

Definition 3.5 (Adjacency Graph)**.** A representation of a graph $G = (V, E)$ by first arbitrarily ordering V into $\mathcal{V}$, then representing E as a matrix $\mathcal{G}$ of dimensions $|V| \times |V|$ and a 1 in $\mathcal{G}_{ij}$ if there is an edge between the ith and jth vertex in $\mathcal{V}$, 0 otherwise. Note that this matrix is inherently symmetrical.

The proof also requires some linear algebra and specifically the Cauchy Interlacing Theorem. We will go over a short proof of the Cauchy Interlacing Theorem here but otherwise assume a basic background in linear algebra. We say that polynomials $f : \mathbb{R} \to \mathbb{R}$ with all real roots $p_0 \leq p_1 \leq ... \leq p_n$ and $g : \mathbb{R} \to \mathbb{R}$ with all real roots $q_0 \leq q_1 \leq ... \leq q_n$ are *interlaced* if $p_0 \leq q_0 \leq p_1 \leq q_1 \leq ... \leq p_n \leq q_n$. f and g are said to be *strictly interlaced* if the inequalities are strict. We take without proof the following theorem (an extension of the Hermite-Biehler Theorem) [10]:

Theorem 3.1 (Hermite-Kakeya [10])**.** *Let P and Q be non-constant polynomials with real coefficients. Then P and Q have strictly interlacing zeros if and only if, for all $\lambda, \mu \in \mathbb{R}$ such that $\lambda^2 + \mu^2 \neq 0$, the polynomial $g(z) := \lambda P(z) + \mu Q(z)$ has simple, real zeros.*

Theorem 3.2 (Cauchy's Interlace Theorem [3])**.** *Let A be a symmetric $n \times n$ matrix, and B be a $m \times m$ principal submatrix of A, for some $m < n$. If the eigenvalues of A are $\lambda_0 \leq \lambda 1 \leq ... \leq \lambda_n$, and the eigenvalues of B are $\mu_0 \leq \mu_1 \leq ... \leq \mu_m$, then for all $0 \leq i < m$*

$$\lambda_i \leq \mu_i \leq \lambda_n$$

That is, their eigenvalues interlace.

Proof. Let $\alpha \in \mathbb{R}$. We can split A into submatrices as follows:

$$A = \begin{bmatrix} B & c^T \\ c & d \end{bmatrix}$$

Then taking the eigenvalues of A to be the roots of the characteristic polynomial $|A - \lambda I| = 0$, and recalling the linearity of the determinant, we can write

$$\begin{vmatrix} B - \lambda I & c^T \\ c & d - \lambda \end{vmatrix} = \begin{vmatrix} B - \lambda I & c^T \\ c & d - \lambda + \alpha \end{vmatrix} - \begin{vmatrix} B - \lambda I & c^T \\ c & \alpha \end{vmatrix}$$

$$\begin{vmatrix} B-\lambda I & c^T \\ c & d-\lambda+\alpha \end{vmatrix} = \begin{vmatrix} B-\lambda I & c^T \\ c & d-\lambda \end{vmatrix} + \begin{vmatrix} B-\lambda I & c^T \\ c & \alpha \end{vmatrix}$$

Where on the right the roots of the characteristic polynomial of the first term yields the eigenvalues of A and the second term the eigenvalues for B. The matrix on the left is still symmetric and, recalling that symmetric matrices have all real eigenvalues, see that the roots of its characteristic polynomial are all real. We can then apply Theorem 3.1 to get that the roots of the characteristic polynomials on the left (ie the eigenvalues of A and B) interlace. □

This turns out to be pretty much all the background and machinery needed for the body of the proof.

3.2 An Equivalent Problem

Definition 3.6. For a Boolean function f, a real multivariate polynomial $p : \mathbb{R}^n \to \mathbb{R}$ is said to *represent* f if for all input $x \in \{0,1\}^n$ we have $p(x) = f(x)$.

Definition 3.7 (Degree Complexity)**.** The degree complexity measure of a Boolean function f, denoted $deg(f)$, is the degree of the unique polynomial p that represents f exactly.

Gotsman and Linial found a way to frame the Sensitivity Conjecture in terms of graph theory, comparing sensitivity to another complexity measure $deg(f)$ which has been shown to be bounded polynomially by $bs(f) \leq deg(f)^2 \leq D(f) \leq bs(f)^4$ [6]. When we say Q_n we mean the graph constructed by taking the vertices hypercube $C^n = \{-1,1\}^n$ and adding an edge between them if they differ by one component.

Theorem 3.3 (Gotsman and Linial [4])**.** *The following are equivalent for any monotone function $h : \mathbb{N} \to \mathbb{R}$.*

(a) For any induced subgraph G of Q^n with a number of vertices not equal to 2^{n-1}, we have $\max(\Delta(H), \Delta(Q^n - H)) \geq h(n)$

(b) For any Boolean function f, we have $s(f) \geq h(deg(f))$

3.3 The Body

The following solution is the work of Huang. The strategy is to prove the following then apply it to Gotsman and Linial's equivalent formulation of the Sensitivity Conjecture.

Theorem 3.4 (Huang [5])**.** *For every integer $n \geq 1$, let H be an arbitrary $(2^{n-1}+1)$-vertex induced subgraph of Q^n. Then,*

$$\Delta(H) \geq \sqrt{n}$$

Moreover this inequality is tight when n is a perfect square.

To prove 3.6, we start by presenting a few supporting lemmas:

Lemma 3.2 ([5]). *Define a sequence of symmetric matrices $A_1, ..., A_n$ recursively as follows:*

$$A_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$A_n = \begin{bmatrix} A_{n-1} & I \\ I & -A_{n-1} \end{bmatrix}$$

Then A^n has dimensions $2^n \times 2^n$ and has $\sqrt{n}$ as an eigenvalue with multiplicity 2^{n-1} and $-\sqrt{n}$ as an eigenvalue with multiplicity 2^{n-1}.

Proof. Note that from the construction the trace of A_n is 0. We first show by induction that $A_n A_n = nI$. For the base case, we quickly see that $A_1 A_1 = I$. For the inductive step, we first write that

$$A_n A_n = \begin{bmatrix} A_{n-1} & I \\ I & A_{n-1} \end{bmatrix} \begin{bmatrix} A_{n-1} & I \\ I & A_{n-1} \end{bmatrix} = \begin{bmatrix} A_{n-1}^2 + I & 0 \\ 0 & A_{n-1}^2 + I \end{bmatrix}$$

which we get from the rules of performing matrix operations with submatrices, though it is also not difficult to see with traditional row column multiplication. Then applying the inductive hypothesis $A_{n-1}A_{n-1} = (n-1)I$ we quickly see that $A_n A_n = nI$ as desired. For any eigenvector $\mathbf{v}$ (existence from the symmetricity of A_n) of A_n we have that $A_n A_n \mathbf{v} = nI\mathbf{v} = n\mathbf{v}$ which implies that any eigenvalue λ of A_n must have $\lambda^2 = n$ and thus $\lambda = \pm\sqrt{n}$. Recalling that the trace of a matrix is the sum of its eigenvalues, it must be the case that the multiplicity of $\sqrt{n}$ and $-\sqrt{n}$ are equal, and equal to $n/2$, as the trace of A_n is zero. □

Lemma 3.3 ([5]). *Let $H = (V, E)$ be a graph and A' its corresponding adjacency matrix. Let the matrix A be the same as A' but with an arbitrary number of 1 entries changed to have a value of -1 instead. Then $\Delta(H)$ is greater than or equal to the absolute value of the largest eigenvalue of A, denoted λ_n, where $n = |V|$.*

Proof. Let $\mathbf{v}$ be the corresponding eigenvector for λ_n. We then see that, for any element v_i of $\mathbf{v}$ we have from basic row column multiplication:

$$|\lambda_n||v_i| = |\lambda_n v_i| = |(A\mathbf{v})_1| = \left| \sum_{j=0}^{|V|-1} A_{i,j}\mathbf{v}_j \right| \leq \sum_{j=0}^{|V|-1} |A_{i,j}\mathbf{v}_j|$$

From the triangle inequality. We then make this key observation regarding adjacency matrices: Column i of A represents the vertices $\mathcal{V}_i$ is connected to, with a non-zero element at $A_{i,j}$ if there is an edge to $\mathcal{V}_j$. Thus $\sum_{j=0}^{|V|-1} |A_{i,j}|$ is the number of edges connected to $\mathcal{V}_i$. Then, from the definition of ΔH we see that this is at most the degree of H; that is, $\sum_{j=0}^{|V|-1} |A_{i,j}| \leq \Delta(H)$. As $|A_{i,j}| = 1$ we then have that

$$\sum_{j=0}^{|V|-1} |A_{i,j}\mathbf{v}_j| \leq \Delta(H)|v_i|$$

$$|\lambda_n||v_i| \leq \Delta(H)|v_i|$$

dividing by $|v_i|$ on both sides yields the desired result. □

Lemma 3.4. *If we take A'_n to be A_n but taking the absolute value of every element, A'_n is exactly adjacency matrix for Q^n, for an explicit arrangement of its vertices.*

Proof. We proceed by induction. For the base case $n = 1$ (ie the cube is a line segment between two points), observe that as the vertices of a cube do not have edges to themselves, so as they don't in Q^n. Thus the trace of the adjacency matrix should have a trace of 0 and have 1s everywhere else. A'_1 fulfills this. For the inductive case, we observe that we can construct a cube of n dimensions by taking two cubes of $n-1$, arbitrarily connecting vertices between them such that each vertex of one $n-1$ dimensional cube connects exactly with one vertex of the other. We can assign an order to these such that in the adjacency graph of Q^n we can append this sorted set of vertex to the $\mathcal{V}$ of Q^{n-1} to form a submatrix $I_{2^{n-1}}$. Each $n-1$ dimensional cube retains their adjacency (sub)matrix from Q^{n-1}. Then by the symmetric nature of adjacency matrices we see that A'_n is an adjacency matrix for Q_n. Thus by induction we have shown the desired result. □

We are now ready to proceed with the proof of Theorem 3.4

Proof. Let A_n be as defined in lemma 3.2. As per lemma 3.4, taking the absolute value of every element yields an adjacency matrix for Q^n. Then for an induced subgraph H of Q^n of $(2^{n-1}+1)$ vertices with adjacency matrix represented by the principle submatrix of A_H of A_n we can apply lemma 3.3, which yields $\Delta(H) \geq \lambda_(2^{n-1}+1)$ where $\lambda_(2^{n-1}+1)$ is the greatest eigenvalue of A_H. Recall from lemma 3.2 that A_n has eigenvalues $\sqrt{n}$ and $-\sqrt{n}$, with multiplicity 2^{n-1} each. As A_H is a $(2^{n-1}+1) \times (2^{n-1}+1)$ dimensional principal submatrix of A_n, we have from Cauchy's Interlace Theorem that (where λ_0 represents the least eigenvalue of A_n, $-\sqrt{n}$)

$$\lambda_{2^{n-1}+1} \geq \lambda_0$$

Putting this all together we have that

$$\Delta(H) \geq |\lambda_{2^{n-1}+1}| \geq |\lambda_0| = |-\sqrt{n}| = \sqrt{n}$$

□

This proof was from [5].

To apply this to the alternative construction of the Sensitivity Conjecture in Theorem 3.3, we take $h(n)$ to be $\sqrt{n}$, observe that for any induced subgraph H with number of vertices 2^{n-1} either H or $Q^n - H$ has $2^{n-1}+1$ vertices, on the greater size of which we apply Theorem 3.4 then extend with lemma 3.1 to see that $\max(\Delta(H), \Delta(Q^n - H)) \geq h(n) = \sqrt{n}$ as desired. Then from Theorem 3.3 we have also shown that for any Boolean function f, we have $s(f) \geq h(deg(f)) = \sqrt{deg(f)}$ and thus $s(f)^2 \geq deg(f)$, and from the bounding $bs(f) \leq deg(f)^2 \leq D(f) \leq bs(f)^4$ we have that $bs(f) \leq s(f)^4$. As such, we have proven the Sensitivity Conjecture. □

We can now be satisfied that these multitudes of complexity measures of Boolean functions are all within polynomial bound of each other.

References

[1] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[2] H. Burman and R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Comptuer Science*, pages 21–43, 2002.

[3] S. Fisk. A very short proof of cauchy's interlace theorem for eigenvalues of hermitian matrices, 2005.

[4] C. Gotsman and N. Linial. The equivalence of two problems on the cube. *J. Combin. Theory Ser.*, pages 142–146, 1992.

[5] H. Huang. Induced subgraphs of hypercubes and a proof of the sensitivity conjecture, 2019.

[6] R. Karthikeyan, S. Sinha, and V. Patil. On the resolution of the sensitivity conjecture. *Bulletin of the American Mathematical Society*, page 1, Mar 2020.

[7] C. Kenyon and S. Kutin. Sensitivity, block sensitivity, and ℓ-block sensitivity of boolean functions. *Information and Computation*, pages 43–53, 2004.

[8] N. Nisan. Crew prams and decision trees. *ACM Press*, pages 327–335, 1989.

[9] N. Nisan and M. Szegedy. On the degree of boolean functions as real polynomials. *Comput.Complexity*, pages 462–467, 1992.

[10] Q. I. Rahman and G. Schmeisser. *Analytic Theory of Polynomials*. Clarendon Press, Oxford, 2002.

[11] H.-U. Simon. A tight $\Omega(\log \log n)$-bound on the time for parallel ram's to compute nondegenerated boolean functions. *Fundamentals (or Foundations) of Computation Theory, Lecture Notes in Computer Science*, 1983.