A COMMUNICATION GAME RELATED TO THE SENSITIVITY CONJECTURE

JUSTIN GILMER, MICHAL KOUCKÝ, AND MICHAEL SAKS

Abstract. One of the major outstanding foundational problems about boolean functions is the *sensitivity conjecture*, which (in one of its many forms) asserts that the degree of a boolean function (i.e. the minimum degree of a real polynomial that interpolates the function) is bounded above by some fixed power of its sensitivity (which is the maximum vertex degree of the graph defined on the inputs where two inputs are adjacent if they differ in exactly one coordinate and their function values are different). We propose an attack on the sensitivity conjecture in terms of a novel two-player communication game. A lower bound of the form $n^{\Omega(1)}$ on the cost of this game would imply the sensitivity conjecture.

To investigate the problem of bounding the cost of the game, three natural (stronger) variants of the question are considered. For two of these variants, protocols are presented that show that the hoped for lower bound does not hold. These protocols satisfy a certain monotonicity property, and (in contrast to the situation for the two variants) we show that the cost of any monotone protocol satisfies a strong lower bound.

There is an easy upper bound of $\sqrt{n}$ on the cost of the game. We also improve slightly on this upper bound.

1. Introduction

1.1. **A Communication Game.** The focus of this paper is a somewhat unusual cooperative two player communication game. The game is parameterized by a positive integer n and is denoted G_n. Alice receives a permutation $\sigma = (\sigma_1, \ldots, \sigma_n)$ of $[n] = \{1, \ldots, n\}$ and a bit $b \in \{0, 1\}$ and sends Bob a message (which is restricted in a way that will be described momentarily). Bob receives the message from Alice and outputs a subset J of $[n]$ that must include σ_n, the last element of the permutation. The cost to Alice and Bob is the size of the set $|J|$.

The message sent by Alice is constrained as follows: Alice constructs an array $\mathbf{v}$ consisting of n cells which we will refer to as *locations*, where each location v_ℓ is initially empty, denoted by $v_\ell = *$. Alice gets the input as a data stream $\sigma_1, \ldots, \sigma_n, b$ and is required to fill the cells of $\mathbf{v}$ in the order specified by σ. After receiving σ_i for $i < n$, Alice fills location σ_i with 0 or 1; once written this can not be changed. Upon receiving σ_n and b, Alice writes b in location σ_n. The message Alice sends to Bob is the completed array in $\{0, 1\}^n$.

A protocol Π is specified by Alice's algorithm for filling in the array, and Bob's function mapping the received array to the set J. The cost of a protocol $c(\Pi)$ is the maximum of the output size $|J|$ over all inputs $\sigma_1, \ldots, \sigma_n, b$.

For example, consider the following protocol. Let $k = \lceil \sqrt{n} \rceil$. Alice and Bob fix a partition of the locations of $\mathbf{v}$ into k blocks each of size at most k. Alice fills $\mathbf{v}$ as follows: When σ_i arrives, if σ_i is the last location of its block to arrive then fill the entry with 1 otherwise fill it with 0.

[0] A preliminary version of this paper appeared in the Proceedings of the 6th Innovations in Theoretical Computer Science conference, 2015.

Supported by NSF grant CCF 083727.

Supported in part by (FP7/2007-2013)/ERC Consolidator grant LBCAD no. 616787 and the project 14-10003S of GA ČR..

Supported by NSF grants CCF-083727, CCF-1218711 and by Simons Foundation award 332622.

Notice that if $b = 1$ then the final array $\mathbf{v}$ will have a single 1 in each block. If $b = 0$ then $\mathbf{v}$ will have a unique all 0 block.

Bob chooses J as follows: if there is an all 0 block, then J is set to be that block, and otherwise J is set to be the set of locations containing 1's. It is clear that $\sigma_n \in J$ and so this is a valid protocol. In all cases the size of J will be at most k and so the cost of the protocol is $\lceil \sqrt{n} \rceil$. We will refer to this protocol as the AND-OR protocol. In Section 2.1 we remark on this protocol's connection to the boolean function

$$\text{AND-OR}(x) = \bigwedge_{i=1}^{\sqrt{n}} \bigvee_{j=1}^{\sqrt{n}} x_{ij}.$$

Let us define $C(n)$ to be the minimum cost of any protocol for G_n. We are interested in the growth rate of $C(n)$ as a function of n. In particular, we propose:

Question 1. *Is there a $\delta > 0$ such that $C(n) = \Omega(n^\delta)$?*

1.2. **Connection to the Sensitivity Conjecture.** Why consider such a strange game? The motivation is that the game provides a possible approach to the well known *sensitivity conjecture* from boolean function complexity.

Recall that the sensitivity of an n-variate boolean function f at an input $\mathbf{x}$, denoted $s_{\mathbf{x}}(f)$, is the number of locations ℓ such that if we flip the bit of $\mathbf{x}$ in location ℓ then the value of the function changes. (Alternatively, this is the number of neighbors of $\mathbf{x}$ in the hamming graph whose f value is different from $f(\mathbf{x})$.) The sensitivity of f, $s(f)$, is the maximum of $s_{\mathbf{x}}(f)$ over all boolean inputs $\mathbf{x}$.

The degree of a function f, $deg(f)$, is the smallest degree of a (real) polynomial p in variables $x_1, \ldots, x_n$ that agrees with f on the boolean cube.

Conjecture 2. *(The Sensitivity Conjecture) There is a $\delta > 0$ such that for any boolean function f, $s(f) \geq \Omega(deg(f)^\delta)$.*

An easy argument (given in Section 2) connects the cost function $C(n)$ of the game G_n to the sensitivity conjecture:

Proposition 3. *For any boolean function on n variables, $s(f) \geq C(deg(f))$.*

In particular, an affirmative answer to Question 1 would imply the sensitivity conjecture.

We note that Andy Drucker [5] independently formulated the above communication game, and observed its connection to the sensitivity conjecture.

1.3. **Background on the Sensitivity Conjecture.** Sensitivity and degree belong to a large class of complexity measures for boolean functions that seek to quantify, for each function f, the amount of knowledge about individual variables needed to evaluate f. Other such measures include decision tree complexity and its randomized and quantum variants, certificate complexity, and block sensitivity. The value of such a measure is at most the number of variables. There is a long line of research aimed at bounding one such measure in terms of another. For measures a and b let us write $a \leq_r b$ if there are constants C_1, C_2 such that for every total boolean function f, $a(f) \leq C_1 b(f)^r + C_2$. For example, the decision tree complexity of f, $D(f)$, is at least its degree $deg(f)$ and thus $deg \leq_1 D$. It is also known [11] that $D \leq_3 deg$. We say that a is *polynomially bounded* by b if $a \leq_r b$ for some $r > 0$ and that a and b are *polynomially equivalent* if each is polynomially bounded by the other.

The measures mentioned above, with the notable exception of sensitivity, are known to be polynomially equivalent. For example, Nisan and Szegedy [13] proved $bs(f) \leq_2 deg(f)$, and also proved a result in the other direction, which was improved in [2] to $deg(f) \leq_3 bs(f)$. For a survey of such results, see [3] and [8]; some recent results include [1, 6].

The sensitivity conjecture, posed as a question by Nisan [12] asserts that $s(f)$ is polynomially equivalent to at least one (and therefore all) of the other measures mentioned. There are many reformulations and related conjectures; see [8] for a survey.

The sensitivity conjecture perhaps more commonly appears as a question on the relationship between sensitivity and block sensitivity. For example, Nisan and Szegedy [13] asked specifically if $bs(f) = O(s^2(f))$ for all functions, and as of this writing no counterexample has been given. The best known bound relating sensitivity to another measure was given by Kenyon and Kutin [9]. They proved that $bs(f) \leq \frac{e}{2\pi} e^{s(f)} \sqrt{s(f)}$ for all boolean functions.

1.4. **Outline of the Paper.** In Section 2 we prove that a positive answer to Question 1 would imply the sensitivity conjecture. We show that adversary arguments for proving that boolean functions are evasive (that is have decision tree complexity $D(f) = n$) provide strategies for the communication game. We also prove that it suffices to answer Question 1 for the restricted class of *order oblivious protocols.*

In Section 3 we present three stronger variants of Question 1. We exhibit protocols that show that two of these variants have negative answers. One might then expect that variants of one of these protocols might lead to a negative answer to Question 1. However, we observe that these protocols satisfy a property called monotonicity and in Section 4 we prove a $\lfloor\sqrt{n}\rfloor$ lower bound on the cost of any monotone protocol. Thus a protocol that gives a negative answer to Question 1 must look quite different from the two protocols that refuted the strengthenings. We also prove a rather weak lower bound for a special class of protocols called assignment oblivious protocols. Finally, in Section 5 we construct a protocol with cost $.8\sqrt{n}$, thus beating the AND-OR protocol by a constant factor. Let $r(k) = \log(C(k)/\log(k))$. After a preliminary version of our paper appeared, Szegedy [14] showed that for any k, $C(n) = O(n^{r(k)})$. Our example shows that there is a k for which $r(k) < 1/2$ and so it follows that $C(n) = O(n^{1/2-\delta})$ for some $\delta > 0$. Szegedy further showed that $C(30) \leq 5$ which gives the best currently known upper bound $C(n) = O(n^{0.4732})$.

2. Connection between the Sensitivity Conjecture and the Game

In this section we prove Proposition 3, which connects the sensitivity conjecture with the two player game described in the introduction.

We use $\mathbf{e}_\ell$ to denote the assignment in $\{0,1\}^n$ that is 1 in location ℓ and 0 elsewhere. Given $\mathbf{v}, \mathbf{w} \in \{0,1\}^n$, $\mathbf{v} \oplus \mathbf{w}$ denotes their bitwise mod-2 sum.

Alice's strategy maps the permutation-bit pair (σ, b) to a boolean array $\mathbf{v}$ and Bob's strategy maps the array $\mathbf{v}$ to a subset of $[n]$. We now show that for each strategy for Alice there is a canonical best strategy for Bob. For a permutation σ, $\Pi_A(\sigma)$ denotes the array Alice writes while receiving $\sigma_1, \cdots, \sigma_{n-1}$ (so location σ_n is labeled $*$). Thus $\Pi_A(\sigma)$ can be viewed as an edge in the *hamming graph* $\mathbb{H}_n$ whose vertex set is $\{0,1\}^n$, with two vertices adjacent if they differ in one coordinate. The *edge set* $E(\Pi)$ of a protocol Π is the set of edges $\Pi_A(\sigma)$ over all permutations σ. This defines a subgraph of $\mathbb{H}_n$. Given Alice's output $\mathbf{v}$, the possible values for σ_n are precisely those locations ℓ that satisfy $(\mathbf{v}, \mathbf{v} \oplus \mathbf{e}_\ell)$ is an edge in $E(\Pi)$. Thus the best strategy for Bob is to output this set of locations. It follows that $c(\Pi)$ is equal to the maximum vertex degree of the graph $E(\Pi)$.

Proposition 3 will therefore follow by showing the following: Given a boolean function with degree n and sensitivity s, there is a strategy Π for Alice for the game G_n such that the graph $E(\Pi)$ has maximum degree at most s.

We need a few preliminaries. A *subfunction* of a boolean function f is a function g obtained from f by fixing some of the variables of f to 0 or 1. For a subfunction g of f, $s(f) \geq s(g)$. We say a function has *full degree* if $deg(f)$ is equal to the number of variables of f. We start by recalling some well known facts.

Lemma 4. *For any boolean function f there exists a subfunction g on $deg(f)$ variables that has full degree.*

Proof. If p is the (unique) multilinear real polynomial that agrees with f on the boolean cube, then p contains a monomial $\prod_{\ell \in S} x_\ell$ where $|S| = deg(f)$. Let g be the function obtained by fixing the variables in $[n] \setminus S$ to 0. Then g is a function on $deg(f)$ variables that has full degree. □

Lemma 5. *Given a function f with full degree and a location ℓ, there exists a bit b such that the function obtained from f by fixing $x_\ell = b$ is also of full degree.*

Proof. The polynomial (viewed as a function from $\{0,1\}^n \to \{0,1\}$) for f may be written in the form $p_1(x_1, x_2, \cdots, \cancel{x_\ell}, \cdots, x_n) + x_\ell p_2(x_1, x_2, \cdots, \cancel{x_\ell}, \cdots, x_n)$. Here $p_1(x_1, x_2, \cdots, \cancel{x_\ell}, \cdots, x_n)$ indicates that the variable x_ℓ is not an input to the polynomial. If p_1 has a non zero coefficient on the monomial $\prod_{k \neq \ell} x_k$, then we set $x_\ell = 0$ and the resulting function will have full degree. For the other case, note p_2 must have a non zero coefficient on $\prod_{k \neq \ell} x_k$ because f has full degree. Thus, setting $x_\ell = 1$ will work. □

The proof of this lemma is essentially the same as the standard argument that the decision tree complexity of any function f is at least $deg(f)$.

We are now ready to prove Proposition 3.

Proof. Given f, let g be a subfunction on $deg(f)$ variables with full degree. We construct a protocol Π that satisfies $E(\Pi) \subseteq E(g)$, where $E(g)$ denotes the set of sensitive edges for the function g, i.e. the edges of $\mathbb{H}_n$ whose endpoints are mapped to different values by g, which implies $c(\Pi) \leq s(g) \leq s(f)$, and thus proves the proposition. As Alice receives $\sigma_1, \sigma_2, \cdots, \sigma_n$, she fills in $\mathbf{v}$ so that the restriction of f to each partial sucessive partial assignment remains a full degree function, which is possible by Lemma 5. After Alice fills location σ_{n-1}, the function g restricted to $\mathbf{v}$ is a non-constant function of one variable, and so the edge $\Pi_A(\sigma)$ is a sensitive edge for g. This implies that $E(\Pi) \subseteq E(g)$.

□

The proof shows that a degree n Boolean function having sensitivity s can be converted into a strategy for Alice for the game G_n of cost at most s. We don't know whether this connection goes the other way, i.e., we can't rule out the possibility that the answer to Question 1 is negative (there is a very low cost protocol for G_n) but the sensitivity conjecture is still true.

2.1. **Connection to Decision Tree Complexity.** An n-variate boolean function is *evasive* if its decision tree complexity is n. A common method for proving evasiveness is via an *adversary argument.* View the problem of evaluating the function by a decision tree as a game between the querier who wishes to evaluate the function and who decides which variable to read next, and the adversary who decides the value of the variable. A function is evasive if there is a strategy for the adversary that forces the querier to ask all n quesitons. For example, to prove that

$$\text{AND-OR}(\mathbf{x}) = \bigwedge_{i=1}^{\sqrt{n}} \bigvee_{j=1}^{\sqrt{n}} x_{ij}$$

is evasive, the adversary can use the strategy: answer 0 to every variable unless the variable is the last variable in its $\bigvee$-block, in which case answer 1. This adversary is exactly Alice's part of the AND-OR protocol described in the introduction. For more examples of adversary arguments see [10].

Every evasive function f by definition admits an adversary argument, and this corresponds to a protocol Π for Alice. In fact a function f is evasive if and only if there exists a protocol Π for which $E(\Pi) \subseteq E(f)$ (recall $E(f)$ is the set of sensitive edges of the function f), and thus the cost (size of the set chosen by Bob) is at most the sensitivity of f. This work explores whether we can use the structure of an arbitrary adversary (or protocol) to exhibit a lower bound on sensitivity.

2.2. **Order Oblivious Protocols.** In the game G_n, at each step $i < n$, the value written by Alice at location σ_i may depend on her knowledge up to that step, which includes both the sequence $\sigma_1, \cdots, \sigma_i$ and the partial assignment already made to v at locations $\sigma_1, \ldots, \sigma_{i-1}$. A natural way to restrict Alice's strategy is to require that the bit she writes in location σ_i depends only on σ_i and the current partial assignment to v but not on the order in which $\sigma_1, \ldots, \sigma_{i-1}$ arrived. A protocol satisfying this restriction is said to be *order oblivious*. The following easy proposition shows that it suffices to answer Question 1 for order oblivious protocols.

Proposition 6. *Given any protocol* Π *there exists an order oblivious protocol* Π' *such that* $E(\Pi') \subseteq E(\Pi)$. *In particular,* $c(\Pi') \leq c(\Pi)$.

Proof. First some notation. Given a permutation σ let $\sigma_{\leq k}$ denote the prefix of the first k elements of σ. We let $\Pi_A(\sigma_{\leq k})$ denote the partial assignment written on $\mathbf{v}$ after Alice has been streamed $\sigma_1, \cdots, \sigma_k$.

Given Π we define an order oblivious protocol Π' of cost at most that of Π. We define Π' in steps, (where in step i Alice receives σ_i and writes a bit in that location). Given $k \geq 0$ we assume that Π' has been defined up through step k and has the property that for every permutation σ, there is a permutation τ of $\sigma_1, \cdots, \sigma_k$ so that $\Pi_A(\tau) = \Pi'_A(\sigma_{\leq k})$.

Suppose σ_{k+1} arrives and the current state of the array is $\mathbf{v} := \Pi'(\sigma_{\leq k})$. From $\mathbf{v}$ Alice can deduce the set $\{\sigma_1, \ldots, \sigma_k\}$ (the set of locations not labeled *). Alice then considers all permutations τ of $\sigma_1, \cdots, \sigma_k$ such that $\Pi_A(\tau) = \Pi'_A(\sigma_{\leq k})$ and picks the lexicographically smallest permutation (call it τ^*) in that set and writes on location σ_{k+1} according to what Π does after τ^*. Note that the bit written on location σ_{k+1} does not depend on the relative order of $\sigma_1, \sigma_2, \cdots, \sigma_k$.

By construction, Π' is order oblivious. Also for any permutation σ there is a permutation τ for which $\Pi_A(\tau) = \Pi'_A(\sigma)$. This implies that $E(\Pi') \subseteq E(\Pi)$. □

3. Stronger Variants of Question 1

We now present three natural variants of Question 1, and refute two of them by exhibiting and analyzing some specific protocols.

The cost function $c(\Pi)$ of a protocol is the worst case cost over all choices of $\sigma_1, \ldots, \sigma_n, b$. Alternatively, we can consider the average size (with respet to random σ and b) of the set Bob outputs. We call this the *expected cost of* Π and denote it by $\tilde{c}(\Pi)$. Let $\tilde{C}(n)$ denote the minimum expected cost of a protocol for G_n.

Question 7. *Is there a* $\delta > 0$ *such that* $\tilde{C}(n) = \Omega(n^\delta)$*?*

An affirmative answer to this question would give an affirmative answer to Question 1.

It is well known that the natural probabilistic version of the sensitivity conjecture, where sensitivity is replaced by average sensitivity (wiith respect to the uniform distribution over $\{0,1\}^n$) is trivially false (for example, for the OR function). However, there is apparently no connection between average sensitivity and average protocol cost. For example, the protocol induced by the decision tree adversary for OR has Alice write a 0 at each step. Note that $E(\Pi)$ is exactly the set of sensitive edges for the OR function. However, the average cost $\tilde{c}(\Pi)$ is $n/2$ whereas the average sensitivity of the OR function is $o(1)$.

We also remark that an analog of Proposition 6 holds for the cost function $\tilde{c}(\Pi)$, and therefore it suffices to answer the question for order oblivious protocols. (The proof of the analog is similar to the proof of Proposition 6, except when modifying the protocol τ^* is not selected to be the lexicographically smallest permutation in the indicated set, but rather the permutation in the indicated set that minimizes the expected cost conditioned on the first k steps.)

There is another natural variant of Question 1 based on average case. When we run a fixed protocol Π on a random permutation σ and bit b, we can view the array $\mathbf{v}$ produced by Alice as a random variable. Let $\tilde{h}(\Pi)$ be the conditional entropy of σ_n given $\mathbf{v}$; intuitively this measures the average number of bits of uncertainty that Bob has about σ_n after seeing $\mathbf{v}$. It is easy to show that this is bounded above by $\log(c(\Pi))$. Let $\tilde{H}(n)$ be the minimum of $\tilde{h}(\Pi)$ over all protocols Π for G_n. The analog of Question 1 is whether there is a positive constant δ such that $\tilde{H}(n) = \Omega(\delta \log(n))$. An affirmative answer to this would have implied an affirmative answer to Question 1, but the answer to this new question turns out to be negative.

Theorem 8. *There is an order oblivious protocol* Π *for* G_n *such that* $\tilde{h}(\Pi) = 3 + \lceil \log\log(n) \rceil$.

Remark: It might seem that this could be proved by giving a protocol Π that is not order oblivious and converting it into an order oblivious protocol as described earlier.. However, while we know that this can be done without increasing worst case cost or average cost, it is possible that $\tilde{h}$ may inrease. Therefore, we construct the desired order oblivious protocol directly.

Proof. Let $k = \lceil \log(n) \rceil$ and associate each location $\ell \in [n]$ to its binary expansion, viewed as a vector $\mathbf{b}(\ell) \in \mathbb{F}_2^k$. Note that $0 \notin [n]$, and thus each vector $\mathbf{b}(\ell)$ is nonzero. For an array $\mathbf{v} \in \{0,1\}^n$ we define $\Gamma(\mathbf{v})$ to be $\sum_{i=1}^n \mathbf{b}(\ell)$, i.e. the vector in $\mathbb{F}_2^k$ obtained by summing the vectors corresponding to the 1 entries of $\mathbf{v}$. Say that an array $\mathbf{v} \in \{0,1,*\}^n$ is *admissible* if there is a way of filling in the *'s (a *completion*) so that for the resulting array $\mathbf{w}$ we have $\Gamma(\mathbf{w}) = 0^k$, where 0^k is the all 0 vector in $\mathbb{F}_2^k$. For an admissible array $\mathbf{v}$, let $\hat{\mathbf{v}}$ be the unique completion of $\mathbf{v}$ such that (1) $\Gamma(\hat{\mathbf{v}}) = 0^k$, (2) The number r of 0's in $\hat{\mathbf{v}}$ is minimum, (3) the ordered sequence $\ell_1 < \cdots < \ell_r$ of locations of the 0's in $\hat{\mathbf{v}}$ is lexicographically minimum subject to conditions (1) and (2), i.e., for each $j \in [r]$, ℓ_j is minimum possible given $\ell_1, \ldots, \ell_{j-1}$.

We now describe the protocol. Let $t > k$ be an integer (which we'll choose to be $\lceil \log^2(n) \rceil$). Alice says 0 for the first $n - t$ steps. The resulting array $\mathbf{u}$ has $n - t$ 0's and t $*$'s. Since $\mathbf{u}$ can be completed to the all 0 array, $\mathbf{u}$ is admissible. Furthermore, among the $*$ positions there must be a set of at most k vectors that sum to 0^k, so $\hat{\mathbf{u}}$ has at most k 1's. Alice fills in the remaining positions to agree with $\hat{\mathbf{u}}$. This strategy is order oblivious: a simple induction shows that for each array $\mathbf{w}$ reached under the above strategy, $\mathbf{w}$ is admissible and $\hat{\mathbf{w}} = \hat{\mathbf{u}}$, so Alice's strategy is equivalent to filling position σ_k (for $k \geq n - t$) according to $\hat{\mathbf{w}}$ where $\mathbf{w}$ is the array after $k - 1$ steps. This is clearly an order oblivious strategy

Let $\mathbf{v}$ denote the array in $\{0,1\}^n$ received by Bob. We now obtain an upper bound on the conditional entropy of σ_n given $\mathbf{v}$. Let $\mathbf{u} = \mathbf{u}(\sigma)$ be the array obtained after the first $n - t$ steps and let $T(\sigma)$ be the set of positions of *'s in $\mathbf{u}$. Let $S(\sigma)$ be the subset of $T(\sigma)$ consisting of those positions set to 0 in $\hat{\mathbf{u}}$. Let L be the random variable that is 1 if $\sigma_n \in S(\sigma)$ and 0 otherwise. Since $S(\sigma)$ depends only on the set $T(\sigma)$ and not on the order of the last t locations, the probability that $L = 1$ is $|S(\sigma)|/|T(\sigma)| \leq \log(n)/\log^2(n) = 1/\log(n)$. We have:

$$\begin{aligned}
H(\sigma_n|\mathbf{v}) &\leq H(\sigma_n, L|\mathbf{v}) \\
&= H(L|\mathbf{v}) + H(\sigma_n|\mathbf{v}, L) \\
&\leq 1 + H(\sigma_n|\mathbf{v}, L) \\
&= 1 + H(\sigma_n|\mathbf{v}, L=1)\Pr[L=1] \\
&\quad +H(\sigma_n|\mathbf{v}, L=0)\Pr[L=0] \\
&\leq 1 + H(\sigma_n)\frac{1}{\log(n)} + H(\sigma_n|\mathbf{v}, L=0)
\end{aligned}$$

We bound the final expression. $H(\sigma) = \log(n)$ so the second term is 1. For the third term, we condition further on the value of the final bit b:

$$H(\sigma_n|\mathbf{v}, L=0) \quad \leq \quad H(b) + \frac{1}{2}(H(\sigma_n|\mathbf{v}, L=0, b=1) + H(\sigma_n|\mathbf{v}, L=0, b=0))$$

Of course, $H(b) = 1$. Given $L = 0$, we have $\sigma_n \in T(\sigma) - S(\sigma)$. If $b = 1$, then σ_n is one of at most t positions set to 1, and so the conditional entropy of σ_n is at most $\log(t) = 2\lceil\log\log(n)\rceil$. If $b = 0$ then $\Gamma(\mathbf{v}) = \sigma_n$ (since σ_n is the unique location that if set to 1 would make the vectors corresponding to the locations of 1's sum to 0^k). The conditional entropy in this case is 0.

Summing up all of the conditional entropy contributions gives $3 + \lceil\log\log(n)\rceil$.

□

For our last variant, suppose Alice can communicate to Bob with a w-ary alphabet instead of a binary alphabet. Thus, Alice is streamed a permutation σ, and when σ_i arrives she may write any of the symbols $\{1, \dots, w\}$ on location σ_i in $\mathbf{v}$. At the last step $b \in \{1, \dots, w\}$ arrives and Alice must write it in location σ_n. Bob sees $\mathbf{v}$ and has to output a set J that contains σ_n. The cost of the protocol is the maximum size of J over all σ and b.

We will show that Question 1 is false in this setting. To state our result we need some definitions. Fix $r > 1$ and positive integer k_0. For $n \geq k_0$ define for each integer $j \geq 0$ the function t_j defined on integers $n \geq k_0$. The function t_0 is given by $t_0(n) = n$ for all n. For $j \geq 1$, t_j is defined inductively $t_j(n) = \max(k_0, \lceil\log_r(t_{j-1}(n))\rceil)$. Observe that for $j \geq 2$ we have $t_j(n) = t_{j-1}(t_1(n)) = t_1(t_{j-1}(n))$. Thus t_j depends on parameters r and k_0 and is a minor variant of the base r iterated log function.

Theorem 9. *For each $j \geq 0$ there is a protocol Π_j using the alphabet $\{1, \dots, 2j+1\}$ that has cost at most $t_j(n)$, where the parameters needed to define t_j are $r = 2^{1/4}$ and some sufficiently large k_0.*

For example, for a ternary alphabet the cost of the protocol is $O(\log(n))$ and for a 5-ary alphabet the cost is $O(\log\log(n))$. To prove this, we'll need a few elementary standard facts about error correcting codes. We include proofs to make the presentation self-contained.

Proposition 10. *For each $n \geq 2$ there is a coloring χ_n of the subsets of $[n]$ by the set $[n^2]$ such that any two sets that have symmetric difference at most 2 get different colors.*

Proof. Construct the graph whose vertices are subsets of $[n]$ with two vertices joined by an edge if their symmetric difference has size 1 or 2. The degree of any vertex is $n(n+1)/2 < n^2$, and so the graph has a proper coloring with color set $[n^2]$. □

If Σ is a finite alphabet and $\mathbf{s} \in \Sigma^k$, a *deletion error* is the removal of some symbol from the string (shrinking the length by 1). We need the following (which is much weaker than what is possible, but is all we need.)

Proposition 11. *There is a k_0 such that for all integers $k \geq k_0$ there is a code C_k of size at least $2^{k/2}$ over $\{0,1\}^k$ that can correct $\lceil 4\log_2(k) \rceil$ deletion errors.*

We note that the k_0 that is needed for Theorem 9 will be the k_0 provided by this Proposition.

Proof. We can choose C_k to be a maximal independent set in the graph on $\{0,1\}^k$ in which two strings $\mathbf{x}$ and $\mathbf{y}$ are joined if there is a string $\mathbf{z}$ that can be obtained from each of them by at most $\lceil 4\log_2(k) \rceil$ deletions. If Δ is the maximum degree of the graph then any maximal independent set has size at least $2^k/(\Delta+1)$ and Δ is at most $\binom{k}{\lceil 4\log_2(k)\rceil}^2 2^{\lceil 4\log(k)\rceil}$ (since given $\mathbf{x}$ each neighbor $\mathbf{y}$ of $\mathbf{x}$ can be constructed by selecting the subset of $\lceil 4\log_2(k) \rceil$ positions to delete from $\mathbf{x}$, the subset of $\lceil 4\log(k) \rceil$ positions to delete from $\mathbf{y}$ and the values of the bits deleted from y). For sufficiently large k this is at most $2^{k/2}-1$. □

Proof of Theorem 9. Fix k_0 according to Proposition 11 and let $r = 2^{1/4}$. Note that $\log_r(n) = 4\log_2(n)$. Define the functions t_j as above.

For $n \leq k_0$ our protocol will just have Alice write the same symbol every time and Bob output $[n]$. So assume $n > k_0$.

We prove the theorem by induction on j. For the induction we need to strengthen the theorem to say that the constructed protocol Π_j works in $j+1$ phases numbered 0 to j where during phase 0, Alice sees $t_0(n) - t_1(n)$ permutation values and writes only $2j+1$ and during phase $i \in [1, j-1]$ Alice processes the next $t_i(n) - t_{i+1}(n)$ permutation values and writes only symbols $2(j-i)+1$ and $2(j-i)+2$. During phase j, Alice processes $t_j(n) - 1$ permutation values and writes symbols 1 and 2.

The protocol Π_0 is trivial: the alphabet is $\{1\}$ and $t_0(n) = n$ and $t_1(n) = 1$. Alice writes only 1's. and Bob outputs the set $[n]$.

Now suppose $j > 1$ and that Π_{j-1} has been defined. Phase 0 of Π_j is prescribed. Let $t = t_1(n)$ and let $S = \{s_1 < \ldots < s_t\}$ be the unfilled positions after phase 0. Alice identifies the set S with the set $[t]$ by the correspondence $s_j \leftrightarrow j$ and views the remaining t symbols of σ as a permutation σ' of $[t]$. The remaining $j-1$ phases of the Π_j correspond to the protocol Π_{j-1} run on σ', so Phase i of Π_j corresponds to Phase $i-1$ of Π_{j-1} run on σ'. For $i \geq 2$, Phase i of Π_j is exactly the same as Phase $i-1$ of Π_{j-1}. However, Phase 1 of Π_j is different from Phase 0 of Π_{j-1}. In Phase 0 of Π_{j-1} the only symbol written is $2j-1$ but in Phase 1 of Π_j both symbols $2j-1$ and $2j$ are used. Since $t \geq k_0$, we can construct C_t as in Proposition 11 and by changing the alphabet, we can view C_t as a subset of $\{2j-1, 2j\}^t$. By the choice of $t = \lceil \log_b n \rceil \geq 4\log_2 n$, we have $n^2 \leq 2^{t/2}$ so we can fix a 1-1 map g from $[n^2]$ to C_t. Alice computes $g(\chi_n(S))$ where χ_n comes from proposition 10. This is a string $\mathbf{y} \in \{2j-1, 2j\}^t$ and during phase 1, Alice write y_i on location s_i. This completes the specification of Π_j.

We now turn to Bob's strategy for choosing the set J to output. Let A_i be the set $\{2i+1, 2i+2\}$. During phase i, Alice only writes symbols from A_{j-i} so the number of symbols from A_{j-i} written by Alice is $d_i(n) = t_i(n) - t_{i-1}(n)$ if $i < j$ and is $d_j(n) = t_j(n) - 1$ if $i = j$. The final symbol b comes from some A_{j-i}; let i^* be the index such that $b \in A_{j-i^*}$.

When receiving Alice's output array Bob can count the number of symbols from each A_{j-i}. For all but one i this will be $d_i(n)$, and will be $1 + d_i(n)$ if and only $i = i^*$.

If $i^* \neq 0$ then Bob knows the set of positions that Alice wrote $2j+1$ to during phase 0, and therefore knows the set S of $t_1(n)$ positions that remained unfilled at the end of phase 0. Since $b < 2j+1$, by identifying symbols $2j$ and $2j-1$, Bob can interpret the array restricted to S as the output of Π_j on a set of size $t_1(n)$. By induction he can determine a set of size at most $t_{j-1}(t(n)) = t_j(n)$ that contains σ_n.

This leaves the case $i^* = 1$ Then Bob sees $n - t_1(n) + 1$ positions that contain $2j+1$ one of which is σ_n. Let S' be the set of positions that don't have $2j+1$ written on them. Then Bob knows S'.

We argue that Bob can recover the set S of positions not written during Phase 0. From this, Bob will know σ_n, since $S - S' = \{\sigma_n\}$.

For those positions $s_i \in S$ that Alice wrote during phase 1, Alice wrote y_i in position s_i where $\mathbf{y} = g(\chi_n(S))$. The number of symbols written during phase 1 is $t_1(n) - t_2(n) = t - \lceil 4\log_2(t) \rceil$ (unless $j = 1$ in which case $t-1$ symbols were written in phase 1). Thus the string $\mathbf{z}$ seen by Bob (using symbols from $\{2j-1, 2j\}$) is obtained from $\mathbf{y}$ with at most $\lceil 4\log_2(t) \rceil$ symbols deleted. Since C_t is robust against $\lceil 4\log_2(t) \rceil$ deletions, Bob can recover $\mathbf{y}$ from $\mathbf{z}$. He then knows $g^{-1}(\mathbf{y}) = \chi_n(S)$. The choice of χ_n implies that S is uniquely determined from S' and $\chi_n(S)$, so Bob recovers S and therefore σ_n. □

4. Lower Bounds for Restricted Protocols

In the previous section, two stronger variants of Question 1 turned out to have negative answers, which may suggest that Question 1 also has a negative answer. In this section however, we prove a lower bound which implies that any counterexample to Question 1 will need to look quite different from the two protocols provided in the last section.

An order oblivious protocol can be specified by a sequence of maps $A_1, \cdots, A_n$ where each A_i maps partial assignments on the set $[n]$ to a single bit. When location σ_i arrives, the bit Alice writes is $A_{\sigma_i}(\mathbf{v})$. For partial assignments α and β, we say that β is an *extension* of α, denoted as $\beta \geq \alpha$, if β is obtained from α by fixing additional variables. An order oblivious protocol is *monotone* if each of the maps $A_1, \cdots, A_n$ are monotone with respect to the extension partial order. That is, if $\beta \geq \alpha$ are partial assignments, then $A_i(\beta) \geq A_i(\alpha)$ for each i. As a remark, when running the protocol there may be assignments that are never written on $\mathbf{v}$, however defining each A_i to have domain all partial assignments is still valid and simplifies notation.

Both the AND-OR protocol described in the introduction and the protocol constructed in Theorem 8 are examples of monotone protocols. Monotonicity generalizes to protocols on w-ary alphabets, and the w-ary protocol of Theorem 9 is monotone (if we order the alphabet symbols in reverse $2j+1 < 2j < \cdots < 1$). Our main result in this section is that monotone protocols on binary alphabets have cost at least $\lfloor \sqrt{n} \rfloor$. In particular, Question 1 is true for such protocols. For the rest of the paper, all protocols will be on binary alphabets.

Theorem 12. *All monotone protocols have cost at least $\lfloor \sqrt{n} \rfloor$.*

Proof. Let Π be a monotone protocol. We show that $E(\Pi)$ has a vertex of degree at least $\lfloor \sqrt{n} \rfloor$.

For a permutation σ denote by $\text{bump}_k(\sigma)$ the permutation obtained from σ by "bumping" the element k to the end of σ and maintaining the same relative order for the rest of σ. For example, $\text{bump}_1(321654) = 326541$.

We let $w(\sigma)$ denote the array $\Pi_A(\sigma)$ with the entries sorted in σ order. In other words, $w(\sigma)$ is the array defined by $w(\sigma)_i = \Pi_A(\sigma)_{\sigma_i}$.

Claim 13. *Let σ be any permutation and let τ be obtained from σ by performing some sequence of bumps on σ. Suppose that τ and $m < n$ satisfy the following:*

- *The elements $\tau_1, \tau_2, \cdots, \tau_m$ were never bumped.*
- *Alice originally wrote a 0 on the locations $\tau_1, \cdots, \tau_m$, that is $\Pi_A(\sigma)_{\tau_i} = 0$ for all $i \leq m$.*

Then $\Pi_A(\tau)_{\tau_i} = 0$ for all $i \leq m$, i.e., $w(\tau)$ begins with m 0's.

Proof. The claim follows easily by induction on i. Suppose we have already shown that $w(\tau)$ begins with $(i-1)$ 0's. Let $\mathbf{v}(\sigma, k)$ denote the partial assignment written on $\mathbf{v}$ just before Alice receives the index k (here the reader should take care to distinguish this from the partial assignment just before Alice receives σ_k). Consider the partial assignment $\mathbf{v}(\tau, \tau_i)$. It follows from the first assumption and the inductive hypothesis that $\mathbf{v}(\sigma, \tau_i)$ is an extension of $\mathbf{v}(\tau, \tau_i)$. Thus, since Alice originally wrote a 0 on location τ_i, by monotonicity she continues to write a 0 on that location when being streamed τ (that is $\Pi_A(\tau)_{\tau_i} = 0$). □

Let σ be the permutation such that $w(\sigma)$ is lexicographically minimum.

Claim 14. *$w(\sigma)$ consists of a block of 0's, followed by a block of 1's, followed by a single *.*

Proof. The result is trivial if their are no 1's. Let j be the location of the first 1, and let k be the last position in the block of 1's beginning at j. We claim $k = n - 1$. Suppose $k < n - 1$. Then there is a 0 in position $k + 1$. Let τ be obtained from σ by bumping $\sigma_j, \ldots, \sigma_k$. By Claim 13, $w(\tau)$ begins with j 0's, contradicting the lexicographic minimality of σ. □

Let $n-t$ be the number of initial 0's in $w(\sigma)$ so the number of 1's is$t-1$. Let $T = \{\sigma_{n-t+1}, \ldots, \sigma_n\}$ and let x be the vector that is 1 in those positions and 0 elsewhere. For k between 1 and n, let $\tau^{(k)} = \text{bump}_k(\sigma)$, so $\tau^{(\sigma_n)} = \sigma$.

The vectors of the form $\Pi_A(\phi)$ and $w(\phi)$ have a single *. For $b \in \{0, 1\}$ we write $\Pi_A(\phi, b)$ and $w(\phi, b)$ for the vectors obtained by replacing the * by b.

Claim 15. *The vertices $\Pi_A(\tau^{(k)}, 1)$ for $k \in T$ are all equal to x. Therefore x belongs to an edge in direction k for each $k \in T$ and so has degree at least t in $E(\Pi)$.*

Proof. Let $k \in T$. Clearly $w(\tau^{(k)}, 1)$ has the first $n-t$ bits 0, and so by the choice of σ the remaining bits are 1. This implies $\Pi_A(\tau^{(k)}$ has 1's in the positions indexed by the last t elements of $\tau^{(k)}$ which is the set T. □

To conclude the proof of the theorem we will find an assignment y that has degree at least $(n - t)/(t + 1)$ in the graph $E(\Pi)$.

Claim 16. *For k among the first $n - t$ elements of σ, $w(\tau^{(k)})$ has the first $n - t - 1$ bits equal to 0, and has at most one 0 among the next t bits (and last bit *).*

Proof. Claim 13 immediately implies that the first $n-t-1$ bits of $w(\tau^{(k)})$ are 0. Now take all of the locations that are labeled 1 in $\Pi_A(\tau^{(k)})$ and bump them to the end and let this new permutation be ρ. Claim 13 implies that all 0's remain 0. By the lexicographic minimality of $w(\sigma)$, $w(\rho)$ has at most $n - t$ 0's which implies that there was at most a single 0 in $\tau^{(k)}$ in positions $n - t + 1$ or higher. □

Now classify each of the first $n-t$ elements of σ into sets $S_{n-t}, \ldots, S_n$. Element $k \in S_n$ if $w(\tau^{(k)})$ has t 1's. Otherwise $k \in S_j$ where j is the location of the unique 0 of $w(\tau^{(k)})$ in locations $n - t$ to $n - 1$. Choose j^* so that $|S_{j^*}|$ is maximum and let $m = |S_{j*}|$, which is at least $(n - t)/(t + 1)$. For $k \in S_{j^*}$, let $y^{ki)} = \Pi_A(\tau^{(k)}, 0)$. Let $u = \sigma_{j^*+1}$ and let y be the vector that is 1 on the positions of $T - \{u\}$ and 0 elsewhere.

Claim 17. *The assignments $y^{(k)}$ for $k \in S_{j^*}$ are all equal to y, and thus y has degree at least m in $E(\Pi)$.*

Proof. By the definition of the bump operation the sequence of elements appearing in positions $n - t, \ldots, n - 1$ in $\tau^{(k)}$ is $\sigma_{n-t+1}, \ldots, \sigma_n$ and the element in position j^* of $\tau^{(k)}$ is $\sigma_{j^*+1} = u$. Thus $y^{(k)}$ is 1 on the elements of $T - \{u\}$ and 0 elsewhere. □

We thus have a point x of degree at least t and a point y of degree at least $(n - t)/t + 1$ in $E(\Pi)$. This implies that cost of Π is at least $\max(t, (n-t)/(t+1)) > \sqrt{n} - 1$ and is thus at least $\lfloor \sqrt{n} \rfloor$. □

As demonstrated by the AND-OR protocol, Theorem 12 is tight up to a constant factor. We remark that the monotone protocols we consider here seem to have no general connection to the class of monotone boolean functions, and our result for monotone protocols seems to be unrelated to the easy and well known fact that the sensitivity conjecture is true for monotone functions.

We conclude this section with a lower bound for a second class of protocols. Although the lower bound is only logarithmic, proving a logarithmic lower bound for all protocols with a large enough constant would improve on the best known bounds relating degree and sensitivity.

We need a few definitions. Recall that an edge $e \in \mathbb{H}_n$ may be written as an array in $\{0,1,*\}^n$ for which $e_\ell = *$ on exactly one location ℓ. We call this location ℓ the *free location* of that edge. We say two edges e, e' *collide* if $e_\ell = e'_\ell$ for all ℓ that is not a free location of either edge. Equivalently, two edges collide if they share at least one vertex (each edge collides with itself). Both of the lower bounds in this section will follow by finding an edge $e \in E(\Pi)$ that collides with m other edges in $E(\Pi)$. This implies at least one of the vertices in e has degree at least $m/2$ in the graph $E(\Pi)$, which in turn lower bounds the cost of the protocol.

For a permutation σ, we write $\ell <_\sigma k$ to denote that the element ℓ comes before the element k in σ. Let $S_k(\sigma) = \{\ell : \ell <_\sigma k\}$. For example, if $\sigma = 321654$ then $S_1(\sigma) = \{2,3\}$. We say a protocol is *assignment oblivious* if the bit written by Alice in location k only depends on the set $S_k(\sigma)$ (and not on the assignment of bits to that set). Such protocols can be described by a collection of n hypergraphs $H_1, H_2, \cdots, H_n$, where each H_ℓ is a hypergraph with vertex set $[n] \setminus \{\ell\}$. When k arrives, Alice writes a 1 if and only if the set $S_k(\sigma)$ is in H_k.

Theorem 18. *Every assignment oblivious protocol* Π *has* $c(\Pi) \geq \log_2(n)/2$.

Proof. Let Π be an assignment oblivious protocol.

Given a permutation $\sigma = \sigma_1\sigma_2\cdots\sigma_n$ and $k \in [n]$ we define $\mathrm{swap}_k(\sigma)$ to be the permutation obtained by swapping the positions of the elements k and σ_n within σ and keeping every other element in the same place. For example, $\mathrm{swap}_3(654321) = 654123$. The lemma will follow by constructing a permutation σ such that that $\Pi_A(\sigma)$ and $\Pi_A(\mathrm{swap}_k(\sigma))$ collide for each $k \in \{\sigma_{n-1}, \cdots, \sigma_{n-\lceil \log_2(n) \rceil}\}$

We build up such a σ in a greedy manner. We start with setting $\sigma_{n-1} = 1$. With σ_{n-1} fixed, the bit Alice writes in location 1 is completely determined by σ_n (and does not depend on the values we later choose for $\sigma_1, \cdots, \sigma_{n-2}$). This holds by the assignment oblivious property and because $S_1(\sigma) = \{\ell : \ell \neq 1, \sigma_n\}$. Let R_1 be the locations ℓ for which setting $\sigma_n = \ell$ results in Alice writing a 1 in location 1. At least one of $|R_1|, |R_1^c|$ are bigger than $\lceil (n-1)/2 \rceil$, let T_1 be that set. Now we fix σ_{n-2} to be any element in T_1.

Having fixed σ_{n-1} and σ_{n-2}, the bit Alice writes on location σ_{n-2} also only depends on the value of σ_n. Now let R_2 be the subset of indices j in T_1 such that setting $\sigma_n = j$ would cause Alice to write a 1 in location σ_{n-2}. At least one of $|R_2|, |R_2^c|$ are bigger than $\lceil (|T_1| - 1)/2 \rceil$, let $T_2 \subseteq T_1$ be that set. This process is iteratively repeated. At step i we set σ_{n-i} to be an arbitrary element of T_{i-1}. With $\sigma_{n-1}, \cdots, \sigma_{n-i}$ now fixed, the value written in location σ_{n-i} depends only on the value of σ_n. The set R_i is defined to be all such values of σ_n that result in Alice writing a 1 in location σ_{n-i} and $T_i \subseteq T_{i-1}$ is defined to be the larger of $|R_i|$ and $|R_i^c|$. We proceed until the set T_i has only one element in it, in this case we assign σ_n to be that element. This process will take at least $\lceil \log_2(n) \rceil$ steps. We then assign the remaining elements to $\sigma_1, \cdots, \sigma_{n-i-1}$ in an arbitrary order.

We now claim that $\Pi_A(\sigma)$ and $\Pi_A(\mathrm{swap}_k(\sigma))$ collide for $k = \sigma_n, \sigma_{n-1}, \cdots, \sigma_{n-\lceil \log_2(n) \rceil}$.

Claim 19. *Let* $i < \lceil \log_2(n) \rceil$, *and let* $k = \sigma_{n-i}$. *Then* $\Pi_A(\sigma)_\ell = \Pi_A(swap_k(\sigma))_\ell$ *for all* $\ell \neq k, \sigma_n$.

Proof. Let $\sigma' = \mathrm{swap}_k(\sigma)$. If $\ell <_\sigma k$ then $S_\ell(\sigma) = S_\ell(\sigma')$ and so Alice writes the same bit to location ℓ under both permutations.

Suppose that $\ell >_\sigma k$. Let j be such that $\sigma_{n-j} = \ell$. Note that $\sigma_{n-1} = \sigma'_{n-1}, \cdots, \sigma_{n-j} = \sigma'_{n-j}$. Recall that holding $\sigma_{n-1}, \cdots, \sigma_{n-j}$ fixed, the bit Alice writes at location ℓ depends only on the value of σ_n, and furthermore that bit is the same as for all settings of $\sigma_n \in T_j$. Since both σ_n and $\sigma'_n = k$ are in the set T_j, it follows that $\Pi_A(\sigma)_\ell = \Pi_A(\sigma')_\ell$. □

By the above claim, σ collides with $\text{swap}_k(\sigma)$ for at least $\lceil \log_2(n) \rceil$ values of k. Furthermore, at least one of the vertices in $\Pi_A(\sigma)$ has degree more than $\lceil \log_2(n)/2 \rceil$. This concludes the proof. □

5. A Protocol with Lower Cost than the AND-OR Protocol

The AND-OR protocol has cost $\lceil \sqrt{n} \rceil$ which matches our lower bound for monotone protocols (within 1). In this section we show that non-monotone protocols can give at least a small advantage:

Theorem 20. *For some $\varepsilon > 0$ and all sufficiently large n there is a protocol Π win $c(\Pi) \leq (1-\varepsilon)\sqrt{n}$.*

Proof. The construction is a variant of the AND-OR protocol.

An (n, m) *proper code* is an indexed family $\{\mathbf{x}_S \in \{0,1\}^n | S \in \binom{[n]}{m}\}$ of vectors such that the support of $\mathbf{x}_S$ is a subset of S. We need the following fact: For n sufficiently large and $n \geq k^2 \geq .8n$ there is an (n, k^2)-proper code in which any two codewords are at hamming distance at least $2k+1$. (The routine proof of this is given below.) Choose the least k such that $k^2 \geq .8n$ and construct such an (n, k^2)-proper code.

Protocol Π is as follows: Alice writes 0 in the first $n - k^2$ locations. Let S be the set of remaining k^2 locations. View S as split into k blocks where the each successive block consists of the smallest k unassigned indices in S. For the last k^2 elements of the permutation, when index j arrives Alice writes $\mathbf{x}_{S,j}$ unless j is the final element of its block to arrive, in which case Alice writes $1 - \mathbf{x}_{S,j}$.

The word received by Bob differs from $\mathbf{x}_S$ in at most k places (one for each block) and so by the distance property of the code, Bob can deduce the set S. If there is a block of S such that the received vector agrees with $\mathbf{x}_S$ on the entire block then Bob outputs that block (since that block must include σ_n); otherwise Bob outputs the set of positions (one per block) in which the received vector disagrees with $\mathbf{x}_S$ (which again must include σ_n).

Finally we prove the existence of the desired (n, k^2)-proper code using a standard random construction. for each $S \in \binom{[n]}{k^2}$ define $\mathbf{x}_S$ to be a random vector supported on S. Call a pair of sets $S, T \in \binom{[n]}{k^2}$ bad if $\mathbf{x}_S$ and $\mathbf{x}_T$ differ in at most $2k+1$ positions. The number of coordinates on which $\mathbf{x}_S$ and $\mathbf{x}_T$ differ is at least the number of coordinates in S on which they differ. Holding $\mathbf{x}_T$ fixed we see that this probability that S, T is bad is at most the probability of fewer than $2k+1$ heads in k^2 coin tosses, which is $2^{-k^2(1-o(1))}$. Taking a union bound over all pairs of k-sets we get that the probability that there is a bad pair is at most $\binom{n}{.2n}^2 2^{-.8n(1-o(1))} = o(1)$, and so with positive probability there are no bad pairs, and so the desired code exists. □

As mentioned in the introduction, after a prelimiary version of this paper appeared, Mario Szegedy [14] gave a protocol of cost $O(n^{.4732})$.

6. Acknowledgements

We thank Ran Raz for helpful discussions. The first author was supported by NSF grant CCF 083727. The second author was supported in part by (FP7/2007-2013)/ERC Consolidator grant LBCAD no. 616787, a grant from Neuron Fund for Support of Science, and the project 14-10003S of GA ČR. The third author was supported by NSF grants CCF-083727 and CCF-1218711, and the Simons Foundation under award 332622.

References

[1] A. Ambainis, K. Balodis, T. Lee, M. Santha, and J. Smotrovs Separations to query complexity based on pointer functions ECCC, TR15-098, 2015.

[2] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. De Wolf. Quantum lower bounds by polynomials. *Journal of the ACM (JACM)*, 48(4):778–797, 2001.

[3] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002.

[4] F. R. Chung, Z. Füredi, R. L. Graham, and P. Seymour. On induced subgraphs of the cube. *Journal of Combinatorial Theory, Series A*, 49(1):180–187, 1988.

[5] A. Drucker Person Communication, 2015.

[6] J. Gilmer, M. Saks, and S. Srinivasan Composition limits and separating examples for some boolean function complexity measures Combinatorica, to appear. (Preliminary version in Proc. 28th IEEE Conference on Computational Complexity, 2013, 185–196; see also arXiv:1306.0630.)

[7] C. Gotsman and N. Linial. The equivalence of two problems on the cube. *Journal of Combinatorial Theory, Series A*, 61(1):142–146, 1992.

[8] P. Hatami, R. Kulkarni, and D. Pankratov. *Variations on the Sensitivity Conjecture.* Number 4 in Graduate Surveys. Theory of Computing Library, 2011.

[9] C. Kenyon and S. Kutin. Sensitivity, block sensitivity, and ℓ-block sensitivity of boolean functions. *Information and Computation*, 189(1):43–53, 2004.

[10] L. Lovasz and N. E. Young. Lecture notes on evasiveness of graph properties. *arXiv preprint cs/0205031*, 2002.

[11] G. Midrijanis. Exact quantum query complexity for total boolean functions. *arXiv preprint quant-ph/0403168*, 2004.

[12] N. Nisan CREW PRAMS and decision trees Proceedings of the twenty-first annual ACM symposium on Theory of computing, 327-335, 1989.

[13] N. Nisan and M. Szegedy. On the degree of boolean functions as real polynomials. *Computational Complexity*, 4:301–313, 1994.

[14] M. Szegedy An $O(n^{0.4732})$ upper bound on the complexity of the GKS communication game ECCC, TR15-102, 2015.

Department of Mathematics, Rutgers University, Piscataway, NJ, USA.
E-mail address: `jmgilmer@math.rutgers.edu`

Computer Science Institute, Charles University, Prague, Czech Republic.
E-mail address: `koucky@iuuk.mff.cuni.cz`

Department of Mathematics, Rutgers University, Piscataway, NJ, USA.
E-mail address: `saks@math.rutgers.edu`