THEORY OF COMPUTING, Volume 13 (7), 2017, pp. 1–18
www.theoryofcomputing.org

A Communication Game Related to the Sensitivity Conjecture

Justin Gilmer* Michal Koucký† Michael Saks‡

Received July 7, 2015; Revised September 12, 2016; Published September 4, 2017

Abstract: One of the major outstanding foundational problems about Boolean functions is the *sensitivity conjecture*, which asserts that the degree of a Boolean function is bounded above by some fixed power of its sensitivity. 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 we show that the cost of any monotone protocol satisfies a strong lower bound in the original variant.

There is an easy upper bound of $\sqrt{n}$ on the cost of the game. We also improve slightly on this upper bound. This game and its connection to the sensitivity conjecture was independently discovered by Andy Drucker (arXiv:1706.07890).

ACM Classification: F.1.3

AMS Classification: 94C10, 68Q17, 68Q15

Key words and phrases: complexity theory, Boolean functions, sensitivity conjecture, sensitivity, degree of Boolean functions, decision tree, communication complexity

A preliminary version of this paper appeared in the Proceedings of the 6th Innovations in Theoretical Computer Science conference, 2015 [7].

*Supported by NSF grant CCF 083727.

†The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 616787. Partially supported by the project 14-10003S of GA ČR and a grant from Neuron Fund for Support of Science.

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

 DOI: 10.4086/toc.2017.v013a007

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.

Definition 1.1. 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.

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

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

Definition 1.2. The function $C(n)$ is defined 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 1.3. 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, which we define now.

Definition 1.4. 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, $s_{\mathbf{x}}(f)$ is the number of neighbors of $\mathbf{x}$ in the hamming graph whose f value is different from $f(\mathbf{x})$.

Definition 1.5. The sensitivity of f, denoted as $s(f)$, is the maximum of $s_{\mathbf{x}}(f)$ over all Boolean inputs $\mathbf{x}$.

Definition 1.6. The degree of a function f, $\deg(f)$, is the smallest degree of a (real) polynomial p in variables $x_1,\ldots,x_n$ such that, for all $\mathbf{x} \in \{0,1\}^n$, $f(\mathbf{x}) = p(\mathbf{x})$.

Conjecture 1.7 (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 1.8. *For any Boolean function on n variables, $s(f) \geq C(\deg(f))$.*

In particular, an affirmative answer to Question 1.3 would imply the sensitivity conjecture.

We note that Andy Drucker [6] independently formulated the above communication game, observed its connection to the sensitivity conjecture, and formulated Question 3.4 given below.

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 [12] 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 [14] proved $\mathrm{bs}(f) \leq_2 \deg(f)$, and also proved a result in the other direction, which was improved in [3] to $\deg(f) \leq_3 \mathrm{bs}(f)$. For a survey of such results, see [4] and [9]; some recent results include [1, 8].

The sensitivity conjecture, posed as a question by Nisan [13] 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 [9] 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 [14] asked specifically if $\mathrm{bs}(f) = O(s^2(f))$ for all functions, and as of this writing no counterexample has been given. The best known bound relating sensitivity to block sensitivity was given by Ambainis et al. [2] who showed that for any Boolean function f, $\mathrm{bs}(f) \le C(f) \le 2^{s(f)-1}s(f)$ (improving on the $\mathrm{bs}(f) \le e^{s(f)+o(1)}$ bound of Kenyon and Kutin [10]).

1.4 Outline of the paper

In Section 2 we prove that a positive answer to Question 1.3 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 1.3 for the restricted class of *order-oblivious protocols*.

In Section 3 we present three stronger variants of Question 1.3. 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 1.3. 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 1.3 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 $\sqrt{0.99n}$, 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 [15] 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 $C(30) \le 5$ 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 1.8, 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.

Definition 2.1. The *hamming graph* $\mathbb{H}_n$ is the unordered graph whose vertex set is $\{0,1\}^n$, where a pair of vertices form an edge if they differ in exactly one coordinate.

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, let $\Pi_A(\sigma)$ denote the array Alice writes while receiving $\sigma_1,\dots,\sigma_{n-1}$ (so location σ_n is labeled $*$). Note that $\Pi_A(\sigma)$ can be viewed as an edge in $\mathbb{H}_n$.

Definition 2.2. The *edge set* $E(\Pi)$ of a protocol Π is the set of edges $\Pi_A(\sigma)$ over all permutations σ.

Definition 2.3. Given a Boolean function f, let $E(f)$ denote the set of sensitive edges of f.

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 1.8 will therefore follow by showing the following: Given a Boolean function with degree d and sensitivity s, there is a strategy Π for Alice for the game G_d 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 2.4. *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 2.5. *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

$$p_1(x_1, x_2, \ldots, x_{\ell-1}, x_{\ell+1}, \ldots, x_n) + x_\ell p_2(x_1, x_2, \ldots, x_{\ell-1}, x_{\ell+1}, \ldots, x_n)\,.$$

If p_1 has a nonzero 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 nonzero 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 1.8.

Proof. Given f, let g be a subfunction of $\deg(f)$ variables with full degree. We construct a protocol Π that satisfies $E(\Pi) \subseteq E(g)$. As Alice receives $\sigma_1, \sigma_2, \ldots, \sigma_n$, she fills in $\mathbf{v}$ so that the restriction of g to each partial assignment remains a full degree function, which is possible by Lemma 2.5. 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 $E(\Pi) \subseteq E(g)$. We conclude that $c(\Pi) \leq s(g) \leq s(f)$. □

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 1.3 is negative 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. It is folklore that every evasive Boolean function admits 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 questions. A common technique for proving a function is evasive is to exhibit an adversary argument. 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 [11].

Adversary arguments correspond to protocols Π for Alice. In fact a function f is evasive if and only if there exists a protocol Π for which $E(\Pi) \subseteq E(f)$. 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 $\sigma_1, \ldots, \sigma_i$ and the partial assignment already made to v at locations $\sigma_1, \ldots, \sigma_{i-1}$. In this section we consider a natural restriction of Alice's strategy.

Definition 2.6. A protocol Π is order-oblivious if, for each i, the bit Alice writes in location σ_i depends only on σ_i and the current partial assignment written on v but not on the order in which $\sigma_1, \ldots, \sigma_{i-1}$ arrived.

The following easy proposition shows that it suffices to answer Question 1.3 for order-oblivious protocols.

Proposition 2.7. *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 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 $\sigma_1, \ldots, \sigma_k$.

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 $\sigma_1, \ldots, \sigma_k$ 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 *). Let τ^* be the lexicographically smallest permutation of $\sigma_1, \ldots, \sigma_k$ such that $\Pi_A(\tau^*) = \Pi'_A(\sigma_{\leq k})$. Alice then 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 $\sigma_1, \sigma_2, \ldots, \sigma_k$.

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 1.3

We now present three natural variants of Question 1.3: Questions 3.1, 3.4 and 3.9. We refute the latter two 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 of the set output by Bob, with respect to uniformly random[1] σ and b, of the set Bob outputs. We call this the *expected cost of* Π and denote it by $\widetilde{c}(\Pi)$. Let $\widetilde{C}(n)$ denote the minimum expected cost of a protocol for G_n.

Question 3.1. Is there a $\delta > 0$ such that $\widetilde{C}(n) = \Omega(n^\delta)$?

An affirmative answer to this question would give an affirmative answer to Question 1.3.

It is well known that the natural probabilistic version of the sensitivity conjecture, where sensitivity is replaced by average sensitivity 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 $\widetilde{c}(\Pi)$ is $(n+1)/2$ whereas the average sensitivity of the OR function is $o(1)$.

We also remark that an analog of Proposition 2.7 holds for the cost function $\widetilde{c}(\Pi)$, and therefore it suffices to answer the question for order-oblivious protocols. The proof of the analog is similar to the proof of Proposition 2.7, 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 information-theoretic variant of Question 1.3. We first formally define some information-theoretic concepts. For a detailed introduction to information theory see [5].

Definition 3.2. The entropy of a random variable X with probability mass function $p(x)$ is

$$H(X) = -\sum_x p(x) \log_2 p(x)\,.$$

Definition 3.3. Given random variables X and Y the conditional entropy of Y given X is

$$H(Y \mid X) = \sum_x p(x) H(Y \mid X = x)\,.$$

When we run a fixed protocol Π on a uniformly random permutation σ and bit b, we can view the array $\mathbf{v}$ produced by Alice as a random variable. Let $\widetilde{h}(\Pi) = H(\sigma_n \mid \mathbf{v})$. 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 $\widetilde{H}(n)$ be the minimum of $\widetilde{h}(\Pi)$ over all protocols Π for G_n. We can now state the information-theoretic analog of Question 1.3, which was independently posed by Andy Drucker [6].

[1] Here we restrict attention to the uniform distribution on σ, b; see the remark following the proof of Theorem 3.5.

Question 3.4. Is there a positive constant δ such that $\widetilde{H}(n) = \Omega(\delta \log(n))$?

An affirmative answer to this would have implied an affirmative answer to Question 1.3, but the answer to this new question turns out to be negative.

Theorem 3.5. *There is an order-oblivious protocol* Π *for* G_n *such that* $\widetilde{h}(\Pi) = 3 + \lceil \log\log(n) \rceil$.

Remark 3.6. 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 $\widetilde{h}$ may increase. 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 $\Gamma(\mathbf{v}) := \sum_{\ell : v_\ell = 1} \mathbf{b}(\ell)$, 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 so that for the resulting array $\mathbf{w}$ we have $\Gamma(\mathbf{w}) = 0^k$, where 0^k is the all-0 vector in $\mathbb{F}_2^k$. We call such a $\mathbf{w}$ a *completion* of $\mathbf{v}$. 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 $\ell_1 < \ldots < \ell_r$ of locations of the 0's in $\hat{\mathbf{v}}$ is lexicographically minimum subject to conditions (1) and (2). That is, 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 $\lceil \log(n) \rceil^2$). 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. Alice fills in the remaining positions to agree with $\hat{\mathbf{u}}$.

This strategy is order-oblivious: A simple induction argument shows that for each array $\mathbf{w}$ reached under the above strategy, $\mathbf{w}$ is admissible and $\hat{\mathbf{w}} = \hat{\mathbf{u}}$.

We also make the following claim.

Claim 3.7. *Let* $S = \{\boldsymbol{b}(\ell) \mid \hat{\boldsymbol{u}}_\ell = 0 \ \& \ \boldsymbol{u}_\ell = *\}$. *We claim that* $|S| \le k$.

Proof. Suppose for contradiction that $|S| \ge k+1$. Then there must be a subset of vectors in S which are linearly dependent. This means we can set those coordinates in $\hat{\mathbf{u}}$ to 1 and still satisfy $\Gamma(\hat{\mathbf{u}}) = 0^k$. This contracts the minimality of $\hat{\mathbf{u}}$. □

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 $\max_\sigma |S(\sigma)|/|T(\sigma)| \le \lceil\log(n)\rceil/\lceil\log(n)\rceil^2 = 1/\lceil\log(n)\rceil$, where the inequality applied Claim 3.7. We have:

$$\begin{aligned} H(\sigma_n \mid \mathbf{v}) &\le H(\sigma_n, L \mid \mathbf{v}) \\ &= H(L \mid \mathbf{v}) + H(\sigma_n \mid \mathbf{v}, L) \\ &\le 1 + H(\sigma_n \mid \mathbf{v}, L) \\ &= 1 + H(\sigma_n \mid \mathbf{v}, L = 1)\Pr[L = 1] \\ &\quad\quad + H(\sigma_n \mid \mathbf{v}, L = 0)\Pr[L = 0] \\ &\le 1 + H(\sigma_n)\frac{1}{\lceil\log(n)\rceil} + H(\sigma_n \mid \mathbf{v}, L = 0)\,. \end{aligned}$$

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

$$H(\sigma_n \mid \mathbf{v}, L = 0) \le H(b) + \frac{1}{2}(H(\sigma_n \mid \mathbf{v}, L = 0, b = 1) + H(\sigma_n \mid \mathbf{v}, L = 0, b = 0))\,.$$

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 $\log(t) \le 2\lceil\log\log(n)\rceil$. 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 $H(\sigma_n \mid \mathbf{v}) \le 3 + \lceil\log\log(n)\rceil$ which concludes the proof. □

Remark 3.8. In Questions 3.1 and 3.4 we assume that the permutation σ is chosen uniformly at random. Other natural variants arise if we allow the adversary to pick an arbitrary distribution. Also, one can consider variants in which Alice and Bob use a randomized protocol and the selected permutation is worst case. We leave these variants as the subject of possible future investigations.

For our last variant of Question 1.3, 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 $\{1, \ldots, w\}$ in location σ_i in $\mathbf{v}$. At the last step $b \in \{1, \ldots, w\}$ 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. For $w \ge 2$, $C^{(w)}(n)$ denotes the minimum cost of any protocol Π on n variables where Alice uses w-ary alphabet. This leads us to:

Question 3.9. For each $w \ge 2$ is there a $\delta(w) > 0$ such that $C^{(w)}(n) = \Omega(n^{\delta(w)})$?

An affirmative answer to this question would have implied the sensitivity conjecture. Indeed, instead of considering functions from $\{0,1\}^n$ to $\{0,1\}$ one can consider functions from $\{1, \ldots, w\}^n$ to $\{0,1\}$. All the complexity measures naturally generalize to this setting, and one can also generalize the sensitivity conjecture. The sensitivity conjecture over w-ary alphabet ($w \ge 2$) is equivalent to the original sensitivity conjecture, and an affirmative answer to Question 3.9 would have implied the sensitivity conjecture over w-ary alphabet.

We will show that Question 3.9 has a negative answer for $w > 2$; in fact $C^{(2j+1)}(n)$ is bounded above by a jth-iterated logarithm of n.

Define the function $\tau : \mathbb{N} \longrightarrow \mathbb{N}$ by $\tau(k) = \lceil 4\log_2(k) \rceil$. Let $k_0 = 2^{128}$ and observe that for $k \geq k_0$ we have $32 \leq \tau(k) \leq k/32$. The functions $(t_j : \mathbb{N} \longrightarrow \mathbb{N} : j \geq 0)$ are defined by:

- $t_0(n) = n$ for all n.
- For $j \geq 1$, $t_j(n) = t_{j-1}(\tau(n))$ if $n \geq k_0$ and $t_j(n) = n$ for $n < k_0$.

Thus t_j is essentially the base-r iterated log function for $r = 2^{1/4}$.

The main result of this part is:

Theorem 3.10. *For each $j \geq 0$ there is a protocol Π_j using alphabet $\{1, \ldots, 2j+1\}$ whose cost is at most $t_j(n)$.*

To describe Π_j we'll need a specialized variant of error correcting codes. For $T \subseteq [n]$, and $s \leq |T|$, $\binom{T}{s}$ denotes the set of subsets of T of size s. For sets S, S', $S \triangle S'$ denotes the symmetric difference $(S - S') \cup (S' - S)$. A *(T,s)-map* over alphabet Σ is a family $\alpha = (\alpha_S : S \in \binom{T}{s})$ where each α_S is a function from S to Σ. We say α is *binary* if $|\Sigma| = 2$, and is *k-robust* if:

> For $S_1, S_2 \in \binom{T}{2}$ satisfying $|S_1 \triangle S_2| = 2$, there are at most $s - k - 1$ indices $j \in S_1 \cap S_2$ such that $\alpha_{S_1}(j) = \alpha_{S_2}(j)$.

Lemma 3.11. *For any subset T of $[n]$, $t = |T|$, and $t \geq s \geq \tau(t)$ there is a binary (T,s)-map that is $\lfloor s/32 \rfloor$-robust.*

(The reader may wish to skip the routine proof and go on to the proof of Theorem 3.10.) The proof of the lemma uses *codes that correct deletion errors*. For a string $\mathbf{s}$, a *deletion* is the removal of some symbol from $\mathbf{s}$ (shrinking its length by 1). We say that a code $C \subseteq \Sigma^k$ *corrects t deletions* provided that for all $v \neq w \in C$, if v' is obtained from v by t deletions and w' is obtained from w by t deletions then $v' \neq w'$. (Thus any code word can be recovered uniquely even after t deletions.)

Proposition 3.12. *For all k, there is a binary code $C_k \subseteq \{0,1\}^k$ with $|C_k| \geq 2^{k/2}$ that corrects $\lfloor k/32 \rfloor$ deletions.*

Proof. Let G_k be the graph on $\{0,1\}^k$ in which two strings $\mathbf{x}$ and $\mathbf{y}$ are joined if they have a common substring $\mathbf{z}$ (not necessarily continuous) of length $(31/32)k$. Any independent set of G_k corrects $\lfloor k/32 \rfloor$ deletions. Let C_k be a maximal independent set. Letting Δ_k denote the maximum degree of a vertex in G_k, then (recalling a standard argument) every vertex of $V(G_k) - C_k$ has at least one edge to C_k and so $|V(G_k)| - |C_k| \leq \Delta_k |C_k|$ which implies $|C_k| \geq 2^k/(\Delta_k + 1)$. Thus, it suffices to show $\Delta_k < 2^{k/2}$.

Note that

$$\Delta_k \leq \binom{k}{\lfloor k/32 \rfloor}^2 2^{\lfloor k/32 \rfloor},$$

since for a fixed string $\mathbf{x}$ each neighbor $\mathbf{y}$ can be specified by selecting the subsets of $\lfloor k/32 \rfloor$ positions to delete from $\mathbf{x}$ and from $\mathbf{y}$ and the values of the bits deleted from y. Using the standard bound $\binom{n}{k} \leq 2^{h(k/n)n}$ where $h(\lambda) = \lambda \log_2(1/\lambda) + (1-\lambda)\log_2(1/(1-\lambda))$, the upper bound on Δ_k is less than $2^{k/2}$. □

Proof of Lemma 3.11. For $i \in T$, let $r_T(i)$ denote the *rank of* i *in* T, i. e., the position of i when T is arranged in increasing order. For $S \subseteq T$, $R(S) = \sum_{i \in S} r_T(i)$. We have $R(S) \leq t^2 \leq 2^{s/2}$, since $s \geq 4\log(t)$. The code C_s given by Proposition 3.12 has size at least $2^{s/2}$, so we can assign each $h \in [t^2]$ to a unique codeword $w_s(h)$ (which is a binary string of length s).

For $S \in \binom{T}{s}$ define α_S so that it maps the jth smallest index in S to the jth bit in $w_s(R(S))$.

To check the conclusion, suppose $S_1, S_2 \in \binom{T}{s}$ and $|S_1 \triangle S_2| = 2$, which implies $R(S_1) \neq R(S_2)$. Let

$$J = \{j \in S_1 \cap S_2 : \alpha_{S_1}(j) = \alpha_{S_2}(j)\}.$$

The codewords $w_s(R(S_1))$ and $w_s(R(S_2))$ can be made equal by deleting $s - |J|$ symbols from each codeword. Since C_s corrects $\lfloor s/32 \rfloor$ deletion errors, $|J| \leq s - \lfloor s/32 \rfloor - 1$. □

Proof of Theorem 3.10. If $n < k_0$, then $t_j(n) = n$ and the protocol is trivial: Alice writes 0 in every position and Bob outputs $[n]$. So assume $n \geq k_0$.

We'll first present the proof for $j = 1$, and then generalize. In Π_1, Alice processes the streamed permutation in two phases numbered 0 and 1. In phase 0, Alice writes 0 in the first $n - \tau(n)$ locations of the streamed permutation. Since $n \geq k_0 \geq 32$, Lemma 3.11 implies that there is a 1-robust binary $([n], \tau(n))$-map α using alphabet $\{1,2\}$. The set L_1 of locations that are unfilled after phase 0 has size $\tau(n)$. During phase 1, Alice receives the next $\tau(n) - 1$ locations and for each such location q writes $\alpha_{L_1}(q)$. The final location σ_n is filled by the adversary to be 0, 1 or 2.

Bob outputs his set as follows. Let M_0 be the set of locations in v that have a 0, and M_1 be the set of locations that have a 1 or 2. If $|M_0| = n - \tau(n)$ then the adversary wrote 1 or 2 in the final location and Bob can output M_1, which has size $t_1(n)$. Otherwise $|M_0| = n - \tau(n) + 1$ and the adversary wrote 0 in location σ_n. For $q \in M_0$, let $M_1(q) = M_1 \cup \{q\}$, so $L_1 = M_1(\sigma_n)$. The values written in locations M_1 agree with α_{L_1}. Since α is a 1-robust $([n], \tau(n))$-map and $|M_1| = |L_1| - 1$, it follows that L_1 is determined (among the sets $M_1(q)$) by the values written on M_1. Thus Bob knows M_1 and $L_1 = M_1 \cup \{\sigma_n\}$ and can output $\{\sigma_n\}$. This completes the case $j = 1$.

For the general case, given $n \geq k_0$, consider the infinite sequence $t_0(n), t_1(n), t_2(n), \ldots$. Let h be the minimum of j and the first index i for which $t_i < k_0$. Let $u_0, u_1, \ldots, u_{h+1}$ be the sequence with $u_i = t_i(n)$ for $i \leq h$ and $u_{h+1} = 1$.

Alice's strategy consists of $h+1$ phases numbered 0 to h. During phase i, Alice processes the next $u_i - u_{i+1}$ elements of the permutation and fills in the corresponding locations (as described below). The unique unfilled location after phase h is labeled by the adversary.

Define the sets $L_0 \supseteq \ldots \supseteq L_{h+1}$ where $L_0 = [n]$ and L_i is the set of unfilled locations after phase $i-1$. Thus $|L_i| = u_i$ for $i \in \{0, \ldots, h+1\}$. The set of locations filled during phase i is $L_i - L_{i+1}$.

During phase 0, Alice writes 0 in each received location. For $0 < i \leq h$, during phase i, Alice writes symbols from $\Sigma_i = \{2i-1, 2i\}$. If $u_i < k_0$ (which is only possible for $i = h$), Alice writes $2h$ in each received location. Otherwise $u_i \geq k_0$ and Alice uses Corollary 3.11 to construct a (L_{i-1}, u_i)-map α^i over alphabet Σ_i that is $\lfloor u_i/32 \rfloor$-robust. In phase i, for each $q \in L_i - L_{i+1}$, Alice writes $\alpha^i_{L_i}(q)$ on location q. (Since Alice knows L_{i-1} and L_i at the start of phase i she can construct $\alpha^i_{L_i}$.)

Bob must then select $J \subseteq [n]$. We will ensure that $|J| = u_h$ or $|J| = 1$. Since $u_h < k_0 \leq t_j(n)$ or $u_h = t_j(n)$, the protocol cost is at most $t_j(n)$.

Alice writes exactly $d_i = u_i - u_{i+1}$ symbols from Σ_i. Let i^* be the index such that the final symbol (chosen by the adversary) lies in Σ_{i^*}. Let M_i be the set of locations whose label is in Σ_i. For $i \neq i^*$, $|M_i| = d_i$, and $|M_{i^*}| = d_{i^*} + 1$. Bob knows the numbers d_i and the sets M_i, and can determine i^*.

If $i^* = h$, then Bob outputs M_h, which has size $u_h = 1 + d_h$ and must contain σ_n. So assume $i^* < h$. Let $K = \bigcup_{j \geq i^*+1} M_j$. For $q \in M_{i^*}$, let $K(q) = K \cup \{q\}$. Bob knows K and M_{i^*} and knows that $\sigma_n \in M_{i^*}$, $L_{i^*} = M_{i^*} \cup K$, and $L_{i^*+1} = K(\sigma_n)$. We claim that Bob can reconstruct the set L_{i^*+1}. If so, he can then output $\{\sigma_n\} = L_{i^*+1} - K$, which costs 1.

By Alice's protocol, the symbols written on M_{i^*+1} agree with $\alpha^{i^*+1}_{K(\sigma_n)}$. So it is enough to show that for each $q \in M_{i^*}$, $q \neq \sigma_n$, $\alpha^{i^*+1}_{K(q)}$ disagrees with $\alpha^{i^*+1}_{K(\sigma_n)}$ on some index in M_{i^*+1}.

Recall that α^{i^*+1} is a (L_{i^*}, u_{i^*+1})-map that is $\lfloor u_{i^*+1}/32 \rfloor$-robust. We claim that $\lfloor u_{i^*+1}/32 \rfloor \geq u_{i^*+2}$. If $i^* + 1 < h$ then $u_{i^*+1} \geq k_0$ and $u_{i^*+2} = \tau(u_{i^*+1}) \leq u_{i^*+1}/32$ by the choice of k_0. If $i^* + 1 = h$, then $u_{i^*+2} = 1$, so it suffices that $u_{i^*+1} \geq 32$, which follows from $u_{i^*+1} = \tau(u_{i^*}) \geq \tau(k_0) \geq 32$.

Therefore α^{i^*+1} is u_{i^*+2}-robust. Since $|K(q) \triangle K(\sigma_n)| = 2$, $\alpha_{K(q)}$ and $\alpha_{K(\sigma_n)}$ can agree on at most $u_{i^*+1} - u_{i^*+2} - 1$ indices, while $|M_{i^*+1}| = u_{i^*+1} - u_{i^*+2}$, so $\alpha_{K(q)}$ and $\alpha_{K(\sigma_n)}$ differ on some location of M_{i^*+1}, as required to complete the proof. □

4 Lower bounds for restricted protocols

In the previous section, two stronger variants of Question 1.3 turned out to have negative answers, which may suggest that Question 1.3 also has a negative answer. In this section however, we prove a lower bound which implies that any counterexample to Question 1.3 will need to look quite different from the two protocols provided in the last section.

Monotone Protocols. An order-oblivious protocol can be specified by a sequence of maps $A_1, \ldots, A_n$ 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 $A_1, \ldots, A_n$ 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. For simplicity we assume that these maps are defined on all partial assignments, even though the resulting protocol may never reach every partial assignment.

Both the AND-OR protocol described in the introduction and the protocol constructed in Theorem 3.5 are examples of monotone protocols. Monotonicity generalizes to protocols on w-ary alphabets, and the w-ary protocol of Theorem 3.10 is monotone (if we order the alphabet symbols in reverse $2j+1 < 2j < \ldots < 1$). Our main result in this section is that monotone protocols on binary alphabets have cost at least $\lfloor \sqrt{n} \rfloor$. In particular, Question 1.3 is true for such protocols. For the rest of the paper, all protocols will be on binary alphabets.

Theorem 4.1. *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 $\mathrm{bump}_k(\sigma)$ 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, $\mathrm{bump}_1(321654) = 326541$.

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 4.2. *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 $\tau_1, \tau_2, \ldots, \tau_m$ were never bumped.*
- *Alice originally wrote a 0 on the locations $\tau_1, \ldots, \tau_m$, 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 4.3. *$w(\sigma)$ consists of a block of 0's, followed by a block of 1's, followed by a single *.*

Proof. The result is trivial if there 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 4.2, $w(\tau)$ 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 $\tau^{(k)} = \mathrm{bump}_k(\sigma)$, 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 4.4. *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. □

We will now find an assignment y that has degree at least $(n-t)/(t+1)$ in the graph $E(\Pi)$.

Claim 4.5. *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 4.2 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 4.2 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 $S_{n-t},\dots,S_n$. 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 $y^{(k)} = \Pi_A(\tau^{(k)},0)$. Let $u = \sigma_{j^*+1}$ and let y be the vector that is 1 on the positions of $T-\{u\}$ and 0 elsewhere.

Claim 4.6. *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 $n-t,\dots,n-1$ in $\tau^{(k)}$ is $\sigma_{n-t+1},\dots,\sigma_n$ 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 $\lfloor\sqrt{n}\rfloor$. This proves Theorem 4.1. □

As demonstrated by the AND-OR protocol, Theorem 4.1 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.

Assignment-Oblivious Protocols. 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 $H_1,H_2,\dots,H_n$, 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 4.7. *Every assignment-oblivious protocol Π has $c(\Pi) \geq \log_2(n)/2$.*

Proof. First some 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).

Let Π be an assignment-oblivious protocol. Our proof follows by finding an edge $e \in E(\Pi)$ which collides with $\log_2(n)$ distinct edges. This will imply that one of the vertices in e has degree at least $\log_2(n)/2$.

Given a permutation $\sigma = \sigma_1\sigma_2\ldots\sigma_n$ 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 $k \in \{\sigma_{n-1},\ldots,\sigma_{n-\lceil\log_2(n)\rceil}\}$

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 $\sigma_1,\ldots,\sigma_{n-2}$). 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 $\sigma_{n-1},\ldots,\sigma_{n-i}$ 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 $\sigma_1,\ldots,\sigma_{n-i-1}$ in an arbitrary order.

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

Claim 4.8. *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 $\sigma_{n-1} = \sigma'_{n-1},\ldots,\sigma_{n-j} = \sigma'_{n-j}$. Recall that holding $\sigma_{n-1},\ldots,\sigma_{n-j}$ 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, $\Pi(\sigma)$ collides with $\Pi(\mathrm{swap}_k(\sigma))$ for at least $\lceil\log_2(n)\rceil$ values of k. Thus, 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 prove existence of a non-monotone protocol with cost $(1-\varepsilon)\sqrt{n}$. After a preliminary version of this paper appeared, Mario Szegedy [15] gave a protocol of cost $O(n^{0.4732})$.

Definition 5.1. An (n,k)-proper code is an indexed family

$$\left\{ \mathbf{x}_S \in \{0,1\}^n \mid S \in \binom{[n]}{k^2} \right\}$$

of vectors such that

1. for each $S \in \binom{[n]}{k^2}$, the support of $\mathbf{x}_S$ is a subset of S,

2. for each $S \neq T$, the hamming distance of $\mathbf{x}_S$ and $\mathbf{x}_T$ is at least $2k+1$.

Lemma 5.2. *For n sufficiently large and $n \geq k^2 \geq 0.99n$ there exists an (n,k)-proper code.*

Proof. We prove existence through a 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 less than $2k+1$ 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, the probability that S,T is bad is at most the probability of obtaining fewer than $2k+1$ heads from k^2 unbiased coin tosses, which is $2^{-k^2(1-o(1))}$. Taking a union bound over all pairs of sets in $\binom{[n]}{k^2}$ we get that the probability that there is a bad pair is at most

$$\binom{n}{0.99n}^2 2^{-0.99n(1-o(1))} = o(1)\,,$$

and so with positive probability there are no bad pairs, and so the desired code exists. □

Theorem 5.3. *For some $\varepsilon > 0$ and all sufficiently large n there is a protocol Π with $c(\Pi) \leq (1-\varepsilon)\sqrt{n}$.*

Proof. The construction is a variant of the AND-OR protocol. Let $k = \lceil \sqrt{0.99n} \rceil$. By Lemma 5.2 there exists an (n,k)-proper code $\{\mathbf{x}_S \mid S \in \binom{[n]}{k^2}\}$. Fix such a code.

The 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 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). □

Acknowledgements

We thank Ran Raz for helpful discussions. We are grateful to the anonymous referees whose comments helped us to improve the presentation of the paper.

References

[1] ANDRIS AMBAINIS, KASPARS BALODIS, ALEKSANDRS BELOVS, TROY LEE, MIKLOS SANTHA, AND JURIS SMOTROVS: Separations in query complexity based on pointer functions. In *Proc. 48th STOC*, pp. 800–813. ACM Press, 2016. Version at ECCC. [doi:10.1145/2897518.2897524, arXiv:1506.04719] 3

[2] ANDRIS AMBAINIS, MOHAMMAD BAVARIAN, YIHAN GAO, JIEMING MAO, XIAOMING SUN, AND SONG ZUO: Tighter relations between sensitivity and other complexity measures. In *Proc. 41st Internat. Colloq. on Automata, Languages and Programming (ICALP'14)*, volume 8572 of *LNCS*, pp. 101–113. Springer, 2014. Version at ECCC. [doi:10.1007/978-3-662-43948-7_9, arXiv:1411.3419] 4

[3] ROBERT BEALS, HARRY BUHRMAN, RICHARD CLEVE, MICHELE MOSCA, AND RONALD DE WOLF: Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001. Preliminary version in FOCS'98. [doi:10.1145/502090.502097, arXiv:quant-ph/9802049] 3

[4] HARRY BUHRMAN AND RONALD DE WOLF: Complexity measures and decision tree complexity: A survey. *Theoret. Comput. Sci.*, 288(1):21–43, 2002. [doi:10.1016/S0304-3975(01)00144-X] 3

[5] THOMAS COVER AND JOY THOMAS: *Elements of Information Theory*. John Wiley & Sons, Inc., 2012. 7

[6] ANDREW DRUCKER: A note on a communication game, 2017. [arXiv:1706.07890] 3, 7

[7] JUSTIN GILMER, MICHAL KOUCKÝ, AND MICHAEL SAKS: A new approach to the sensitivity conjecture. In *Proc. 9th Internat. Conf. on Information Communication Technology and Systems (ITCS'15)*, pp. 247–254. ACM Press, 2015. [doi:10.1145/2688073.2688096] 1

[8] JUSTIN GILMER, MICHAEL SAKS, AND SRIKANTH SRINIVASAN: Composition limits and separating examples for some boolean function complexity measures. *Combinatorica*, 36(3):265–311, 2016. Preliminary version in CCC'13. [doi:10.1007/s00493-014-3189-x, arXiv:1306.0630] 3

[9] POOYA HATAMI, RAGHAV KULKARNI, AND DENIS PANKRATOV: *Variations on the Sensitivity Conjecture*. Number 4 in Graduate Surveys. Theory of Computing Library, 2011, pp. 1–27. [doi:10.4086/toc.gs.2011.004] 3

[10] CLAIRE KENYON AND SAMUEL KUTIN: Sensitivity, block sensitivity, and ℓ-block sensitivity of Boolean functions. *Inform. and Comput.*, 189(1):43–53, 2004. [doi:10.1016/j.ic.2002.12.001] 4

[11] LÁSZLÓ LOVÁSZ AND NEAL E. YOUNG: Lecture notes on evasiveness of graph properties, 2002. [arXiv:cs/0205031] 6

[12] GATIS MIDRIJANIS: Exact quantum query complexity for total Boolean functions, 2004. [arXiv:quant-ph/0403168] 3

[13] NOAM NISAN: CREW PRAMs and decision trees. *SIAM J. Comput.*, 20(6):999–1007, 1991. Preliminary version in STOC'89. [doi:10.1137/0220062] 3

[14] NOAM NISAN AND MARIO SZEGEDY: On the degree of Boolean functions as real polynomials. *Comput. Complexity*, 4(4):301–313, 1994. Preliminary version in STOC'92. [doi:10.1007/BF01263419] 3

[15] MARIO SZEGEDY: An $O(n^{0.4732})$ upper bound on the complexity of the GKS communication game, 2015. [arXiv:1506.06456] 4, 15

AUTHORS

Justin Gilmer
Google
gilmer@google.com

Michal Koucký
Associate professor
Charles University, Prague, Czech Republic
koucky@iuuk.mff.cuni.cz
http://iuuk.mff.cuni.cz/~koucky/

Mike Saks
Distinguished professor of mathematics
Rutgers University, Piscataway, NJ
saks@math.rutgers.edu
https://www.math.rutgers.edu/~saks/

ABOUT THE AUTHORS

JUSTIN GILMER graduated from Rutgers in 2015; his advisor was Michael Saks. He lives in Mountain View, California and is a researcher in Machine Learning at Google. He loves hiking, biking, and the free food at work.

MICHAL KOUCKÝ received his Ph. D. in Computer Science from Rutgers in 2003, where he was advised by Eric Allender. After having great time as a postdoc at McGill in Montréal and CWI in Amsterdam, he moved back to his home town—Prague. He spent almost a decade at the Institute of Mathematics of the Czech Academy of Sciences, and then came back to his alma mater, Charles University. He is interested in various aspects of theoretical computer science but he also loves programming. He is always amazed by beauty that can be found in math and nature.

MIKE SAKS received his Ph. D. in Mathematics from M.I.T. in 1980, where he was advised by Daniel Kleitman. He was a postdoctoral fellow at UCLA, and held positions at Bell Communications Research and the Computer Science and Engineering Department at UCSD. He has worked in a variety of areas in theory of computing and discrete mathematics: lower bounds for data structures, circuits, communication complexity, branching programs and decision trees, streaming algorithms, sublinear algorithms, satisfiability algorithms, online algorithms, distributed computing, and derandomization, and extremal problems for graphs, hypergraphs and partially ordered sets.