COMS 6998: Communication Complexity April 20, 2022

Number-on-forehead Communication Complexity

Course instructor: *Toni Pitassi* Author: *Daniel Mitropolsky*

1 Introduction

Throughout these notes, f is a function $f : \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_k \to \mathcal{Z}$. Most often, f will be a function family $f_{n,k} : (\{0,1\}^n)^k \to \{0,1\}$. We could study the *multiparty communication complexity* of computing $f_{n,k}$, the k inputs of length n are divided between k parties. We could try to characterize the communication complexity as a function of n and k. There are in fact two widespread models for multiparty communication:

Definition 1. In the *Number-in-hand (NIH)* model, each player i sees input $x_i \in \mathcal{X}_i$ only.

Definition 2. In the *Number-on-forehead model (NOF)* Player i sees every input $x_j \in \mathcal{X}_j$ for $j \neq i$.

Important: in both these models, the communication is "broadcast"; that is, all communication bits are written on a blackboard for all parties to see.

Definition 3. Just like in 2-party protocols, a k-party protocol Π for f specifies:

(a) which player communicates given the round i and the communication so far;

(b) if player j communication at round i, what bits they write on the blackboard as a function of i, the communication so far, and the *input* (which in the NIH model is x_i, and in the NOF model is x_j for $j \neq i$);

(c) the value of $f(x_1, \ldots, x_k)$ from the communication when the protocol terminates.

Note that an essentially equivalent definition would replace (c) with the requirement that the communication transcript ends with $f(x_1, \ldots, x_k)$.

Now, both the NIH and NOF models are interesting and are studied, but these notes are focused on the **NOF** model.

In these notes we will study two notions of communication complexity, although the analogues of non-deterministic, and public/private-coin randomized communication complexity from the 2-party case also carry over and are also sometimes studied in the NOF model.

Definition 4. The *k-party deterministic communication complexity* of f is the minimum communication over all deterministic protocols computing f and is denoted $D_k(f)$.

Definition 5. The *randomized communication complexity* of f with respect to a distribution μ over $\mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_k$ with error ϵ is the minimum communication over all deterministic protocols computing such that the protocol errs on at most ϵ fraction of the inputs when they are drawn from μ. This is denoted $R_k^{\epsilon,\mu}(f)$.

Note that in the most frequent case, when f is shorthand for a function family $f_{n,k} : (\{0,1\}^n)^k \to \{0,1\}$, $D_k(f)$ and $R_k^{\epsilon,\mu}(f)$ will be functions of k and n; perhaps the notation should take n into account ($D_k(f_n)$ or $D_{n,k}(f)$ for instance) but this should be clear from context (and is consistent with the literature).

1.1 A motivating example

As a motivating example we will consider the function EQ, that is, the function family $EQ_{n,k} : (\{0,1\}^n)^k \to \{0,1\}$ which outputs 1 iff all the k-inputs are equal. What is its multiparty communication complexity, in the NIH and NOF models?

Observation 6. For $k = 2$, the NOF model is equivalent to the NIH model, and both are the same as our normal 2-party communication model. For $k = 2$ we know that $EQ_{n,2}$ requires n bits of communication, so it has maximal communication complexity.

Observation 7. For $k \geq 3$, in the NIH model, the communication complexity is still $\geq n$, since 2 players could easily simulate additional players in order to use a k-party protocol to solve equality.

Now we will see our first example where computation in the NOF model is significantly different from the NIH model:

Claim 8. *For $k \geq 3$, in the NOF model, $D_k(EQ) = 2$.*

Proof. The protocol is as follows: player 1 sends 1 iff all the other parties' inputs are equal, and player 2 sends 1 iff all the other parties' input equal. $EQ(x_1, \ldots, x_k)$ iff both bits are 1. Also, any protocol computing EQ must use at least 2 bits of communication: for a 1-bit protocol where player i sends the bit, we could change only their input in order to change the value of EQ but not the bit sent by player i. □

The "takeway" from this example is that in the NOF model, we can exploit the fact that players see multiple inputs, and that these inputs overlap, to come up with an efficient protocol for certain functions. We could not do this in the NIH model.

Observation 9. Another observation is that the EQ protocol is extremely simple in the following way: the communication of each player does not depend on the communication of any other player– we call such players *oblivious*, and if every player is oblivious, the protocol is "oblivious" or "simultaneous" since they could all write their communication at the same time.

Perhaps since NOF communication complexity is a relatively new field and much is unknown, many of the known protocols in the field are oblivious. In other words, we have rarely used adaptivity based on previous players' communication to do anything clever.

Definition 10. The *simultaneous deterministic communication complexity* of f is the minimum communication over *simultaneous* protocols computing f in the NOF model. Denote this $S_k(n)$.

2 Connection to Circuit Complexity

One reason to care about NOF communication complexity is that it has an important connection to showing *lower bounds in circuit complexity*. Specifically, we know a way to show that f is *not* in a certain

circuit class given that it has high enough NOF communication complexity. However, we have not been able to use this method yet, because we do not have good enough NOF communication complexity lower bounds (*yet*). In this section we will show this (as of yet, unused) method.

Definition 11. $\mathrm{AC}^0[m]$ is the class of languages that can be computed by a family of circuits $\{C_n\}$ such that each $C_n : \{0,1\}^n \to \{0,1\}$ has

(a) constant depth;

(b) size polynomial in n;

(c) the gates are $\{\wedge, \vee, \neg, \mod m\}$, with unbounded fan-in.

In the above definition the $\mod m$ gate has $\log(n)$ output wires. In order to capture languages computable by such circuits with any mod-gate, we have the following complexity class:

Definition 12. $\mathrm{ACC}^0 = \bigcup_{m \geq 2} \mathrm{ACC}^0[m]$

Definition 13. A logical gate is *symmetric* if its output depends only on the number of 1's and 0's in the inputs (i.e., for any permutation of the input wires, the gate computes the same function).

A weaker version of the following theorem was first shown by Yao, but the version presented was proven slightly later by Beigen and Tarui.

Theorem 14. *(Beigel and Tarui '94, [4]) For $L \in \mathrm{ACC}^0$, $\exists\, c, d$ and circuits C_n that compute L which have the following form:*

(a) depth 2;

(b) size at most $2^{\log^d n}$;

(c) the top layer is a single symmetric *gate;*

(d) the bottom layer consists of $\wedge$ gates with fan-in $\log^c n$.

The above theorem is one of the most important characterizations of ACC^0 languages. It is related to the following observation:

Theorem 15. *(Håstad and Goldmann '91) Suppose $f : \{0,1\}^n \to \{0,1\}$ can be computed by circuits C_n with:*

(a) depth 2;

(b) the top gate is symmetric with fan-in s;

(c) the bottom layer consists of $\wedge$ gates with fan-in $\leq k-1$.

Then, under any *partition of n into k parties, $S_k(f) \leq k \log(s)$.*

Proof. Since there are more parties than inputs to each $\wedge$ gate, each $\wedge$ gate can be computed by some party (pigeonhole principle). Partition the gates among parties. Each party communicates how many of its gates evaluate to 1 (this can be done simultaneously; since the top gate has fan-in s there are at most s non trivial $\wedge$-gates so total communication is $k \times \log(s)$). Since the top symmetric gate is determined by how many of its inputs are 1, the value of f is implied by the communication. □

Note that if $D_k(f) \leq S_k(f)$. Combining the two theorems we have the following, where we abuse notation and say a function f is in ACC^0 if the language $L_n = \{x \in \{0,1\}^n \ : \ f(x) = 1\}$ is in ACC^0.

Corollary 16. *For any function f in ACC^0, $\exists\, c, d$ such that under any partition of n bits to $k = \log^c n + 1$ parties, $D_k(f) \leq S_k(f) \leq (\log^c n + 1)\log^d n = \log^{O(1)} n$.*

The corollary above is the "connection" between circuit-complexity and NOF communication complexity. Showing that functions are not in ACC^0 is a major goal in circuit complexity.

Observation 17. If we could show some function $f = f_{n,k}$ with $k = \log^c n + 1$ requires $S_k(f) > \log^{O(1)} n$, this would show $f \notin \mathrm{ACC}^0$.

This method for showing a function is not in ACC^0 has not been successfully applied so far. So far we have not shown a function where $k \geq \log(n)$ and that has a super-logarythmic communication-complexity. In fact, the functions for which we know the best lower bounds are in ACC^0, rather trivially, as we shall see. However this remains a major prospect for showing lower bounds. Separately, we know that $\mathrm{NEXP} \subsetneq \mathrm{ACC}^0$ [9], but that was not shown using this method.

3 Lay of the land

Here is an overview of some of the most important things we know about NOF communication complexity.

3.1 Lower bounds

Our one "good" method ("discrepancy method" + removing cylinder intersections with Cauchy-Schwarz, as it is known in the field) give lower bounds of a similar flavor:

- For the *Generalized Inner Product* (GIP) function, $D_k(\mathrm{GIP}) \geq \Omega(\frac{n}{4^k})$. *This function is defined in section 4, and this result is proven in section 6*
- For the *Disjointness* function, $D_k(\mathrm{DISJ}) \geq \Omega(\frac{n}{4^k})$.

Similar lower bounds are known for a few other related functions. There are also a few lower bounds from other methods, that are much weaker. The most important is

- For the function Exactly-n : $[n]^3 \to \{0,1\}$ (which is studied for $k = 3$), $D_3(\textsc{Exactly}\text{-}n) \geq \Omega(\log\log\log n)$.

3.2 Upper bounds

We also known some surprising and clever algorithms in the NOF model ("clever" like our EQ algorithm was, taking advantage of shared information), giving upper bounds on the communication complexity.

- For the *Generalized Inner Product* (GIP) function, $D_k(\mathrm{GIP}) \leq O(k\frac{n}{2^k})$.
- For the function Exactly$-n : [n]^3 \to \{0,1\}$, which is studied for $k = 3$, $D_3(\textsc{Exactly}\text{-}n) \leq \sqrt{\log n}$. *This result is proven in section 7*

Note that for the *Generalized Inner Product* (GIP) function, the upper-bound and lower-bound almost match. For the Exactly$-n$ function, there is an enormous gap between the clever algorithm achieving communication $O(\sqrt{\log(n)})$ and the lower bound, which is $\log\log\log(n)$.

4 Generalized Inner Product

Definition 18. For $x_1, \ldots, x_k \in (\{0,1\}^n)^k$, $\mathrm{GIP}_{n,k}(x_1, \ldots, x_k) = \bigoplus_{i=1}^n (x_1)_i \wedge \cdots \wedge (x_n)_i$. That is, $\mathrm{GIP}_{n,k}(x_1, \ldots, x_k)$ = number of coordinates that all equal 1, mod 2.

4.1 How do the lower bounds relate to ACC^0?

As mentioned, the method for showing a function is not in ACC^0 via NOF communication complexity has not been used. How exactly is it not applicable given our best lower bounds, such as the one for GIP?

Observation 19. The lower bound for GIP is only non-trivial if $k < \log(n)/2$. For $k \geq \log(n)/2$, the lower bound is $n/4^k = n/4^{2\log(n)} = 1$. For the method of Observation 17 to work, the lower bound would have to be at greater than $\log^{O(1)}(n)$ (in fact, it would also have to work for any $k = \log^{O(1)}(n)$).

However, in the case of GIP, we couldn't hope for such a lower bound. On one hand, we have an almost matching upper bound, but also:

Proposition 20. *Viewing $GIP_{n,k}$ for $k = \log^c n$ and vectors of size n as a function $f_{n'}$ on $n' = n\log^c n$ bits, $f_{n'} \in ACC^0$. In fact, $f_{n'} \in AC^0[2]$.*

Proof. The bottom layer will have n AND-gates, computing $(x_1)_i \wedge \cdots \wedge (x_k)_i$ for each coordinate $i \in [n]$, which feed into a $\mod 2$ gate, which outputs $f_{n'} = GIP_{n,k}$. □

Observation 21. In the previous proof, the circuit showing the function is in $\mathrm{AC}^0[2]$ is, not so coincidentally, in the Beigel-Tarui form of Theorem 14.

5 Basic machinery for NOF communication complexity

Definition 22. A *cylinder* C_i in the i-th coordinate is a subset of the input space $\mathcal{X}_1 \times \cdots \times \mathcal{X}_k$ that does not depend on the i-th coordinate. That is, if $(x_1, \ldots, x_i, \ldots, x_k) \in C_i$ then for all $x_i' \in \mathcal{X}_i, (x_1, \ldots, x_i', \ldots, x_k) \in C_i$.

Definition 23. A *cylinder intersection* C is an intersection of cylinders.

Proposition 24. *If C_i, C_i' are cylinders in the i-th coordinate, so is $C_i \cap C_i'$. Hence, any cylinder intersection C can be written $\cap_{i=1}^k C_i$.*

The proof is a very easy exercise.

Proposition 25. *For a NOF protocol P with communication c, the set of inputs that induce a particular communication transcript $t \in \{0,1\}^c$ is a cylinder intersection.*

Proof. The proof is recursive, bit-by-bit of the communication transcript. It is true before any bit is written (since the entire input space is a cylinder intersection). Inductively, at step i, the set of inputs that results in the (partial) transcript $c_1, \ldots, c_{i-1}$ is a cylinder intersection C. If at step i player j speaks, whether they write bit c_i depends only on the inputs of every *other* player, so the inputs which result in the next bit being c_i is a cylinder C_i in the j-th coordinate. Hence the set of inputs resulting in partial transcript $c_1, \ldots, c_{i-1}c_i$ is $C \cap C_i$, a cylinder intersection. □

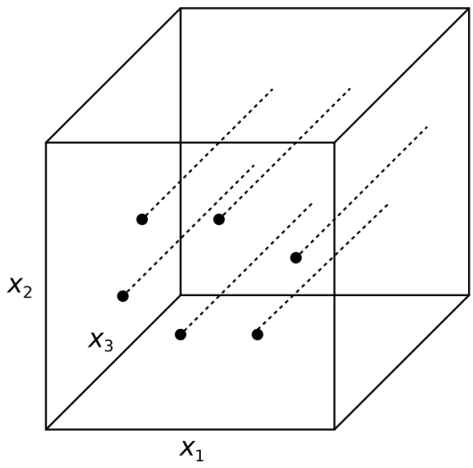

Figure 1: This figure from [1] shows a cylinder in the 3rd coordinate, for $k = 3$. See how it is specified by a subset of the other coordinates (i.e. a subset of coordinate 1 $\times$ coordinate 2)

Corollary 26. *If P is a deterministic NOF protocol computing $f : \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_k \to \mathcal{Z}$ with c bits of communication, P partitions $\mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_k$ into at most 2^c monochromatic cylinder intersections.*

Proof. Because the transcript of P determines the value of f, all inputs resulting in the same transcript must have the same value of f. □

Cylinder intersections are the analogue of *rectangles* from 2-party communication complexity. In fact, it is easy to see that:

Observation 27. For $k = 2$, cylinder interesctions are rectangles. See Figure 2 for a "proof by picture".

Rectangles are still far from fully understood (for instance, their exact connection to the log-rank of a matrix), but cylinder intersections are far more complex combinatorial objects. In many ways, our limited understanding of cylinder intersections are why we have such limited techniques and lower bounds in NOF communication complexity.

5.1 Discrepancy method

The reader may recall the extremely useful "discrepancy method" in 2-party communication complexity. It turns out this method translates exactly the NOF case with k parties. As in the 2-party case, we do the usual trick where we replace the output space of f. From now on, $f : \mathcal{X}_1 \times \cdots \times \mathcal{X}_k \to \{\pm 1\}$.

From now on we will abuse notation, and for a cylinder intersection C, we will write $C(x_1, \ldots, x_k) = 1$ if $x_1, \ldots, x_k \in C$ and 0 otherwise.

Definition 28. For a distribution μ over $\mathcal{X}_1 \times \cdots \times \mathcal{X}_k$, a function $f : \mathcal{X}_1 \times \cdots \times \mathcal{X}_k \to \{\pm 1\}$, and cylinder intersection C, the *discrepancy* of f with respect to μ and C is

$$\mathrm{disc}_\mu(f, C) = \left| \mathop{\mathbb{E}}_{x_1,\ldots,x_k \sim \mu} [f(x_1, \ldots, x_k) C(x_1, \ldots, x_k)] \right|$$

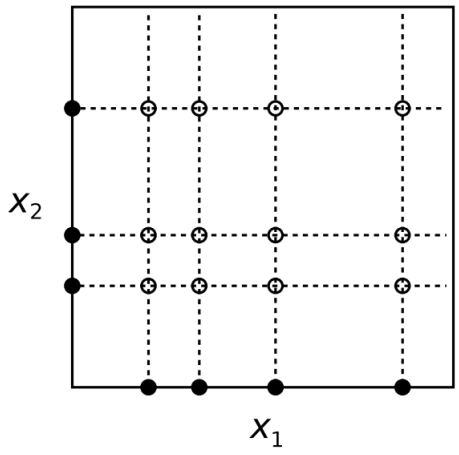

Figure 2: This illustration from [1] shows a cylinder intersection when $k = 2$, which is a rectangle.

Definition 29. The *discrepancy* of f wrt μ is

$$\mathrm{disc}_\mu(f) = \max_C \mathrm{disc}_\mu(f, C)$$

where the max is over cylinder intersections C.

The *intuition* behind the definition of discrepancy is that it is the biggest average value of f over any cylinder intersection. It scales with size of the cylinder intersection. If the value is close to 0 it means that f is "well-spread" over $+1$ and -1 in any cylinder intersection, and if it is large, it means there is a large cylinder intersection that is heavily biased one way. Hence, if the discrepancy is small, we would need many cylinder-intersections to monochromatically cover the input space.

Theorem 30. *(Discrepancy Method; Babai, Nisan, Szegedy '92, [2]) For any $f : \mathcal{X}_1 \times \cdots \times \mathcal{X}_k \to \{\pm 1\}$,*

$$R_k^{\epsilon,\mu} \geq \log\left(\frac{1-2\epsilon}{disc_\mu(F)}\right)$$

Proof. The proof is exactly the same as the one for the discrepancy method for the $k = 2$ cases. In fact it is a good exercise to revisit the original proof and see what it carries over (and exactly where and why rectangles get replaced by cylinder intersections). □

The discrepancy method is how we prove our best known lower bounds. We will show that

Theorem 31. *$disc_U(\mathrm{GIP}) \leq exp(-n/4^k)$, where $U = U_{n,k}$ is the uniform distribution over $(\{0,1\}^n)^k$.*

Given the previous theorem, which is hard, the discrepancy method gives us the desired lower bound:

Theorem 32.

$$R_k^{\epsilon,U}(\mathrm{GIP}) \geq n/4^k + \log(1-2\epsilon)$$

And in particular,

$$D_k(\mathrm{GIP}) \geq n/4^k$$

Proof. This follows immediately from combining Theorems 30 and 31. □

6 The Cube-measure bound for discrepancy

We will need some syntactic sugar for the rest of the proofs:

Definition 33. For **two inputs** $(x_1^0, \ldots, x_k^0)$ and $(x_1^1, \ldots, x_k^1)$ in $(\{0,1\}^n)^k$, and a vector $b \in \{0,1\}^k$, x^b denotes the "mixed input" $(x_1^{b_1}, \ldots, x_k^{b_k})$.

Theorem 34. *(Cube-measure bound for discrepancy) For* any $f : (\{0,1\}^k)^n \to \{\pm 1\}$,

$$disc_U(f)^{2^k} \leq \mathop{\mathbb{E}}_{\substack{(x_1^0,\ldots,x_k^0)\\(x_1^1,\ldots,x_k^1)}} \Big[\prod_{b \in \{0,1\}^k} f(x^b) \Big]$$

This is an extremely important theorem. The "moral" is this: it replaces the *discrepancy* of f with respect to the uniform distribution, a definition that involves expectations over *cylinder intersections*, with an expectation over pairs of inputs of a (somewhat complex) product of values of f. The upper bound is crude, as we shall see, making repeated use of the Cauchy-Schwarz inequality. The point is, however, that the expectation on the right-hand side is much easier to analyze than anything involving cylinder-intersections. For example, we know:

Theorem 35. *(Cube-measure of GIP)*

$$\mathop{\mathbb{E}}_{\substack{(x_1^0,\ldots,x_k^0)\\(x_1^1,\ldots,x_k^1)}} \Big[\prod_{b \in \{0,1\}^k} \mathrm{GIP}(x^b) \Big] \leq e^{-n/2^{k-1}}$$

These two theorems combine to give us Theorem 31:

Proof. $\mathrm{disc}_U(\mathrm{GIP}) \leq (e^{-n/2^{k-1}})^{1/2^k} \leq 2^{-n/4^k}$. □

The two theorems have very different and interesting proofs, but the second one is slightly simpler.

6.1 Cube-measure of GIP

In this section we prove Theorem 35. Recall that we switched the output space of our functions to $\{\pm 1\}$:

$$\begin{aligned} \mathop{\mathbb{E}}_{\substack{(x_1^0,\ldots,x_k^0)\\(x_1^1,\ldots,x_k^1)}} \Big[\prod_{b \in \{0,1\}^k} \mathrm{GIP}(x^b) \Big] &= \mathbb{E}\Big[\prod_{b \in \{0,1\}^k} \prod_{i=1}^{n} (-1)^{x_{1,i}^{b_1} \wedge \cdots \wedge x_{k,i}^{b_k}} \Big] \\ &= \mathbb{E}\Big[\prod_{i=1}^{n} \prod_{b \in \{0,1\}^k} (-1)^{x_{1,i}^{b_1} \wedge \cdots \wedge x_{k,i}^{b_k}} \Big] \end{aligned}$$

Because the inputs are uniform, the coordinates are *independent*, and hence we can move the product out of the expectation:

$$= \prod_{i=1}^{n} \mathbb{E}\Big[\prod_{b \in \{0,1\}^k} (-1)^{x_{1,i}^{b_1} \wedge \cdots \wedge x_{k,i}^{b_k}} \Big]$$

But again because the coordinates are independent, by *symmetry* the expectation inside the product is the same for any coordinate. Hence we get,

$$= \Big(\mathbb{E}_{\substack{x_1^0,\ldots,x_k^0\in\{0,1\}\\ x_1^1,\ldots x_k^1\in\{0,1\}}}\Big[\prod_{b\in\{0,1\}^k}(-1)^{x_0^{b_1}\wedge\cdots\wedge x_k^{b_k}}\Big]\Big)^n$$

Where, since the expectation is only over one coordinate, we replaced the two lists of *vectors* with two lists of *bits*. Now, fix $x_1^0,\ldots,x_k^0$ and $x_1^1,\ldots,x_k^1$ and consider the value of the product:

- If for all $j\in[k]$, $x_j^0\neq x_j^1$, then the product is -1. This is because there is a unique b such that $f(x^b)=-1$.
- If for some j, $x_j^0=x_j^1$, then product is 1. This is because for any b such that $f(x^b)=-1$, b' which is b but flips the bit in position j also has $f(x^{b'})=-1$, so they cancel.

The probability of $x_1^0,\ldots,x_k^0$ and $x_1^1,\ldots,x_k^1$ that are of the first case is exactly $1/2^k$, and the probability of the second case is $1-(1/2^k)$. Hence,

$$\begin{aligned}\Big(\mathbb{E}_{\substack{x_1^0,\ldots,x_k^0\in\{0,1\}\\ x_1^1,\ldots x_k^1\in\{0,1\}}}\Big[\prod_{b\in\{0,1\}^k}(-1)^{x_0^{b_1}\wedge\cdots\wedge x_k^{b_k}}\Big]\Big)^n &= \big((1-1/2^k)-1/2^k\big)^n\\ &= (1-1/2^{k-1})^n\\ &=\leq e^{-n/2^{k-1}}\end{aligned}$$

This concludes the proof of Theorem 35. □

6.2 Proof of cube-measure bound

In this section we prove Theorem 34. We will need:

Lemma 36. *(Cauchy-Schwarz)*

$$\mathbb{E}[Z]^2\leq\mathbb{E}[Z^2]$$

Now, recall that

$$\begin{aligned}\mathrm{disc}_U(f) &= \max_C \mathrm{disc}_U(f,C)\\ &= \max_C \Big|\mathop{\mathbb{E}}_{x_1,\ldots,x_k}[f(x_1,\ldots,x_k)C(x_1,\ldots,x_k)]\Big|\end{aligned}$$

The proof is by induction: assume the theorem statement is true for *any* function with $k-1$ players where the inputs have size n. We will show that it is true for *any* function with k players where inputs have size n (in this way, the theorem is separately true for every n and k).

For $f:(\{0,1\}^n)^k$, let $C=\cap_{i=1}^k C_i$ be the maximizing cylinder intersection for discrepancy, i.e.

$$\mathrm{disc}_U(f)=\Big|\mathop{\mathbb{E}}_{x_1,\ldots,x_k}[f(x_1,\ldots,x_k)\Pi_{i=1}^k C_i(x_1,\ldots,x_k)]\Big|$$

Since C_k does not depend on x_k,

$$= |\mathop{\mathbb{E}}_{x_1,\ldots,x_{k-1}}[C_k(x_1,\ldots,x_{k-1},\cdot)\mathop{\mathbb{E}}_{x_k}[f(x_1,\ldots,x_k)\Pi_{i=1}^{k-1}C_i(x_1,\ldots,x_k)]|$$

By squaring both sides, and applying Cauchy-Schwarz,

$$\begin{aligned}\mathrm{disc}_U(f)^2 &\le \mathop{\mathbb{E}}_{x_1,\dots,x_{k-1}}[(\mathbb{E}_{x_k}[(C_k(x_1,\dots,x_{k-1},\cdot))^2 f(x_1,\dots,x_k)\Pi_{i=1}^{k-1}C_i(x_1,\dots,x_k)])^2] \\ &\le \mathop{\mathbb{E}}_{x_1,\dots,x_{k-1}}[(\mathbb{E}_{x_k}[f(x_1,\dots,x_k)\Pi_{i=1}^{k-1}C_i(x_1,\dots,x_k)])^2]\end{aligned}$$

Where the second inequality is true because $C_k(x_1,\dots,x_{k-1},\cdot) \in \{0,1\}$ and because of the squaring, the inside of the expectation is non-negative. Now, we actually expand the square:

$$\begin{aligned}&= \mathop{\mathbb{E}}_{x_1,\dots,x_{k-1},x_k^0,x_k^1}[f(\dots,x_k^0)f(\dots,x_k^1)\Pi_{i=1}^{k-1}C_i(\dots,x_k^0)C_i(\dots,x_k^1)] \\ &= \mathop{\mathbb{E}}_{x_k^0,x_k^1}[\mathop{\mathbb{E}}_{x_1,\dots,x_{k-1}}[f^{x_k^0,x_k^1}(x_1,\dots,x_{k-1})\Pi_{i=1}^{k-1}C_i^{x_k^0,x_k^1}(x_1,\dots,x_{k-1})]\end{aligned}$$

Above, the last line is just a way to re-write the previous line. That is, we define $f^{x_k^0,x_k^1}(x_1,\dots,x_{k-1}) := f(\dots,x_k^0)f(\dots,x_k^1)$, and $C_i^{x_k^0,x_k^1}(x_1,\dots,x_{k-1}) := C_i(\dots,x_k^0)C_i(\dots,x_k^1)$. But note that (after fixing x_k^0 and x_k^1), $f^{x_k^0,x_k^1}$ is just some function on $k-1$ inputs of size n, and $C_i^{x_k^0,x_k^1}$ is just some cylinder over $k-1$ coordinates (well, the indicator of a cylinder). Hence, for any fixed x_k^0, x_k^1, the inner expectation upper bounded by $\mathrm{disc}_U(f^{x_k^0,x_k^1})$.

Raising both sides of the inequality by 2^{k-1} and applying Cauchy-Schwarz $k-1$ times, we have

$$\mathrm{disc}_U(f)^{2^k} \le \mathop{\mathbb{E}}_{x_k^0,x_k^1}[\Big(\mathop{\mathbb{E}}_{x_1,\dots,x_{k-1}}[f^{x_k^0,x_k^1}(x_1,\dots,x_{k-1})\Pi\ C_i^{x_k^0,x_k^1}(x_1,\dots,x_{k-1})]\Big)^{2^{k-1}}]$$

Applying induction to the inner expectation (separately for each x_k^0, x_k^1, which we can based off how we worded the inductive hypothesis so that it's true for function with $k-1$ parties), we have,

$$\begin{aligned}&\le \mathop{\mathbb{E}}_{x_k^0,x_k^1}\ \mathop{\mathbb{E}}_{\substack{(x_1^0,\dots,x_{k-1}^0)\\(x_1^1,\dots,x_{k-1}^1)}}\Big[\prod_{b\in\{0,1\}^{k-1}} f^{x_k^0,x_k^1}(x^b)\Big] \\ &= \mathop{\mathbb{E}}_{\substack{(x_1^0,\dots,x_k^0)\\(x_1^1,\dots,x_k^1)}}\Big[\prod_{b\in\{0,1\}^{k-1}} f(x^b,x_k^0)f(x^b,x_k^1)\Big] \\ &= \mathop{\mathbb{E}}_{\substack{(x_1^0,\dots,x_k^0)\\(x_1^1,\dots,x_k^1)}}\Big[\prod_{b\in\{0,1\}^{k}} f(x^b)\Big]\end{aligned}$$

This concludes the proof of Theorem 30. □

7 An algorithm for Exactly$-n$

In this section we prove one of the other results listed in the lay-of-the-land of Section 3, namely the upper-bound for Exactly$-n$. The algorithm is beautiful and uses geometric arguments. In the 3-party case, we will refer to the players as Alice, Bob, and Charlie.

Definition 37. Exactly-n is the 3-party function $f : [n]^3 \to \{0,1\}$ where $f(x,y,z) = 1$ iff $x+y+z = n$.

Figure 3: The first coloring is 3-AP free, the second is not.

It is important to remember that the input size here is $\log(n)$. Hence there is a trivial protocol with $\log(n)+1$ bits of communication (Alice sends Bob's input, and Bob outputs f). That is, a trivial upper-bound *linear* in the input size.

Though we did not define communication complexity over protocols that themselves use randomness, we briefly mention it in connection to this problem:

Proposition 38. *If the parties are allowed to use random coins, and must succeed with error at most* $1/3$*, there is a protocol with* $\log\log(n)$ *communication (*logarithmic *in the input size).*

Proof. Charlie computes $z' = n - x - y$, and Alice and Charlie run the 2-party equality protocol with Alice using Charlie's input z and Charlie using z'. Note $x+y+z = n$ iff $z = z'$. Because the random-coin communication complexity of equality is logarithmic in the input size, this protocol uses $\log\log(n)$ bits of communication. □

In fact, Exactly-n is part of a broader category of 3-party functions called *graph* functions. Let $g : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}^l$ be a 2-party function. The graph function of g, $f_g : \{0,1\}^m \times \{0,1\}^m \times \{0,1\}^l \to \{0,1\}$ is the function $f_g(x,y,z) = 1$ iff $g(x,y) = z$. These are well studied because of this additional structure. Exactly-n is the graph function of $g : [n] \times [n] \to [n]$ given by $g(x,y) = n - x - y$. The above proposition and proof clearly applies to *any* graph function, giving a random-coin protocol with communication $\log(l)$ (in our case $l = \log(n)$).

Now, back to deterministic communication complexity. We will show a protocol with communication square-root in the input size, that is

Theorem 39. $S_3(f) \leq \sqrt{\log n}$

Recall that $S_3(f)$ is the *simultaneous* deterministic communication complexity of f, in particular $D_3(f) \leq S_3(f)$. This is the best algorithm at the time of writing; it may be that non-simultaneity (i.e. adaptivity) could be exploited to get a better protocol.

Definition 40. A *coloring* is a mapping from $[n]$ to a color set C. It is "3-AP-free" (AP stands for *arithmetic progression*) if for any sequence $a, a+b, a+2b \in [n]$, they do not have the same color.

See Figure 3 for an example of colorings.

The algorithm is based on the following classic result:

Theorem 41. *(Behrend, 1946 [3]) There is a 3-AP-free coloring of* $[n]$ *with* $2^{O(\sqrt{\log n})}$ *colors.*

First we show how to prove theorem 39 given this theorem:

Proof. Let $x' = n - y - z$, $y' = n - x - z$. Observe $x - x' = y - y' = x + y + z - n$.

Hence, $x + 2y', x' + 2y, x + 2y$ is a 3-AP (with difference $x+y+z-n$ between each consecutive pair).

They are all equal iff $x+y+z = n$. All three numbers are in $[-2n, 3n]$ and can be computed by Bob, Alice, and Charlie, respectively. Using a coloring for $[5n]$ from Theorem 41, each player computes their color in the progression and sends it (simultaneously). Then, $f(x,y,z) = 1$ iff the colors are different. This requires communication $3 \times \log(2^{O(\sqrt{\log(5n)})}) = O(\sqrt{\log n})$ □

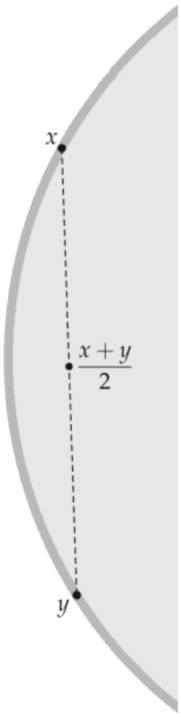

Figure 4: A "proof by picture" of Theorem 42

7.1 Behrend's theorem

This section proves Theorem 41. Observe that another way to write a 3-AP in $[n]$ is as three numbers $x, \frac{x+y}{2}, y \in [n]$.

Proposition 42. *Suppose we had a "homomorphism" from $[n]$ to some vector space $\mathbb{R}^d$, that is, a mapping $v : [n] \to \mathbb{R}^d$ with the property that for $x, \frac{x+y}{2}, y \in [n]$, $v(\frac{x+y}{2}) = \frac{v(x)+v(y)}{2}$. Then, $\|v(x)\|$ would be a 3-AP-free coloring.*

Proof. If we had such a mapping, and $\|v(x)\| = \|v(y)\|$, then $v(x)$ and $v(y)$ are points on the d-sphere of radius $\|v(x)\|$. On the other hand, $\frac{v(x)+v(y)}{2}$ is the midpoint between the vectors and hence, by geometry (the analytic details are omitted), does *not* lie on the same sphere, so its norm (and hence color) cannot be $\|v(x)\|$. See Figure 4 for a "proof by picture" of this idea. □

We might try to construct such a homomorphism the inuitive way. Fix a base d, and fix r such that $d^r > n$. Let $v(x) \in \mathbb{R}^r$ be the base-d representation of x. We would like to say $\|v(x)\|^2$ works as a coloring– if it did, then since $\|v(x)\|^2$ is an integer and has value at most $d^2 r$, we would get away with $d^2 r = O(\log n)$ colors. With the protocol from the proof of Theorem 39, such a coloring would give a $O(\log \log n)$ communication protocol, i.e., logarithmic in the input size Unfortunately, this does not seem to work: using base$-d$ representations of numbers in $[n]$ is not a "homomorphism" in our sense, i.e. it does not guarantee the homomorphic mid-point property of Proposition 42.

The idea of the full construction is to add an extra piece to the coloring (in addition to $\|v(x)\|^2$) which forces the homomorphic mid-point property, and then the geometric argument will apply (we can't have

$\|v(x)\|^2 = \|v(y)\|^2 = \|v(\frac{x+y}{2})\|^2$ if the homomorphic mid-point property holds).

We will need to choose d such that $4|d$, and as before, $d^4 > n$. The coloring will be $\|v(x)\|^2$ *as well as* a vector $w(x) \in \mathbb{R}^d$, called the the *approximation* of $v(x)$: $w(x)_i$ is largest number $jd/4$ for $j \in \{0,1,2,3,4\}$ such that $jd/4 \leq x_i$ (that is, $w(x)_i$ is $v(x)_i$ rounded to the nearest $d/4$).

The color of x is $(\|v(x)\|^2, w(x))$. Observe that there are at most $5^r = 2^{O(r)}$ values for $w(x)$ and, as before, $d^2 r$ for $\|v(x)\|^2$. Overall this gives $2^{O(r)+\log d}$ colors. Using $r = \sqrt{\log n}, d = 2^{\sqrt{\log n}}$ (for which $d^r > n$ as needed), we get $2^{O(\sqrt{\log n})}$ colors.

Finally, for $a, a+b, a+2b \in [n]$, we want to show that if $w(a) = w(a+b) = w(a+2b)$ then we have the "homomorphic midpoint property" of Proposition 42, that is, $v(a+b) = \frac{v(a)+v(a+2b)}{2}$. But then its not possible that $\|v(x)\|^2 = \|v(y)\|^2 = \|v(\frac{x+y}{2})\|^2$ concluding the proof.

For $x \in [n]$ let $W(x)$ be the number represented by $w(x)$ (that is, $\sum_{i=0}^{r} w(x)_i d^i$). By construction of $w(x)$, the base-d representation of $x - W(x)$ is $v(x) - w(x)$. Additionally (and this is where $4|d$ comes in), $2(x - W(x))$ in vector form is $2(v - w(x))$. Hence we have, beginning with a triviality and then using the fact that $w(a) = w(a+b) = w(a+2b)$,

$$\begin{aligned} a + 2b + a &= 2(a+b) \\ a + 2b - W(a+2b) + a - W(a) &= 2(a + b - W(a+b)) \\ v(a+2b) - w(a+2b) + v(a) - w(a) &= 2(v(a+b) - w(a+b)) \\ v(a+2b) + v(a) &= 2v(a+b) \end{aligned}$$

Where the last line is what we wanted to force. □

References

[1] Anil Ada. Notes on communication complexity. Excellent notes, `https://www.cs.mcgill.ca/~rraada/CCnotes.pdf`.

[2] László Babai, Noam Nisan, and Mario Szegedy. Multiparty protocols and logspace-hard pseudorandom sequences (extended abstract). In *STOC 1989*, 1989. Original lower bounds paper.

[3] F. A. Behrend. On sets of integers which contain no three terms in arithmetical progression. *Proceedings of the National Academy of Sciences*, 32(12):331–332, 1946.

[4] Richard Beigel and Jun Tarui. On acc. *Comput. Complex.*, 4(4):350–366, oct 1994.

[5] Troy Lee. Lecture 2: Multiparty number-on-the-forehead complexity, 2012. Excellent notes, `https://www.csc.kth.se/utbildning/kth/kurser/DD2441/semteo12/lecturenotes/NotesLec13.pdf`.

[6] Shachar Lovett. Cse 291: Communication complexity, winter 2019, multi-party protocols, 2019. Excellent notes, `https://cseweb.ucsd.edu/classes/wi19/cse291-b/5-multiparty.pdf`.

[7] Toniann Pitassi. Foundations of communication complexity, lecture 5, 2014. First part covers connection to ACC, `https://www.cs.toronto.edu/~toni/Courses/PvsNP/Lectures/lecture5.pdf`.

[8] Anup Rao and Amir Yehudayoff. *Communication Complexity: and Applications*. Cambridge University Press, 2020. Chapters 4 and 5 in `https://yehudayoff.net.technion.ac.il/files/2016/03/book.pdf`.

[9] Ryan Williams. Nonuniform acc circuit lower bounds. 61(1), jan 2014.