Electronic Colloquium on Computational Complexity, Report No. 94 (2025)

Communication Complexity is NP-hard

Shuichi Hirahara
National Institute of Informatics, Japan

Rahul Ilango
MIT, USA

Bruno Loff
LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

1. In the paper where he first defined Communication Complexity [3], Yao asks: *Is computing* $\mathsf{CC}(f)$ *(the 2-way communication complexity of a given function* f*) NP-complete?* The problem of deciding whether $\mathsf{CC}(f) \leq k$, when given the communication matrix for f and a number k, is easily seen to be in NP. Kushilevitz and Weinreb have shown that this problem is cryptographically hard [2]. Here we show it is NP-hard.

The proof consists of a small collection of observations, on top of a previous reduction by Jiang and Ravikumar [1] from vertex cover. They showed implicitly that it is NP-hard to compute $\chi_1(f)$, the 1-partition number (we will give a formal definition later) of a given function. We will begin by describing their reduction in the language of communication matrices. Using their reduction and a few additional tricks, we will conclude that the communication complexity of f is NP-hard to compute exactly.

2 *Notation and definitions.* Let $f : X \times Y \to \{0,1\}$ be a communication matrix. We let $\mathsf{CC}(f)$ be deterministic communication complexity of f, i.e., the smallest depth of a *binary* protocol tree for computing f.[1]

A *rectangle* (of $f : X \times Y \to \{0,1\}$) is a product set $A \times B$ where $A \subseteq X$ and $B \subseteq Y$. We let $\chi_1(f)$ denote the 1-partition number, i.e., the smallest number of pairwise disjoint 1-monochromatic rectangles needed to cover $f^{-1}(1)$. Because the 1-monochromatic leaves of any communication protocol for f form a partition of $f^{-1}(1)$, we have:

$$\mathsf{CC}(f) \geq \lceil \log \chi_1(f) \rceil.$$

3. We now describe a previous result by Jiang and Ravikumar [1], where they implicitly show that χ_1 is NP-hard. The reduction is via the vertex cover problem. Recall, in the vertex cover problem, we are given as input an undirected graph $G = ([n], E)$, with $E \subseteq \binom{[n]}{2}$, and we wish to find the size $\kappa(G)$ of a smallest-possible set $C \subseteq [n]$ such that $|e \cap C| \geq 1$ for all $e \in E$. Given G, we will now describe a communication matrix $f_G : X \times Y \to \{0,1\}$, such that

$$\chi_1(f_G) = n + 4|E| + \kappa(G).$$

The matrix f_G and the proof that $\chi_1(f) = n + 4|E| + \kappa(G)$ appears as Lemma 3.3. of Jiang and Ravikumar's paper [1]. They actually showed this for the *normal set basis problem*, which turns out to be equivalent to χ_1, and here we translate their reduction to the language of communication complexity.

The communication matrix of f_G consists of $2n + 4|E|$ rows, indexed by the set

$$([n] \times \{0,1\}) \cup (E \times [4]),$$

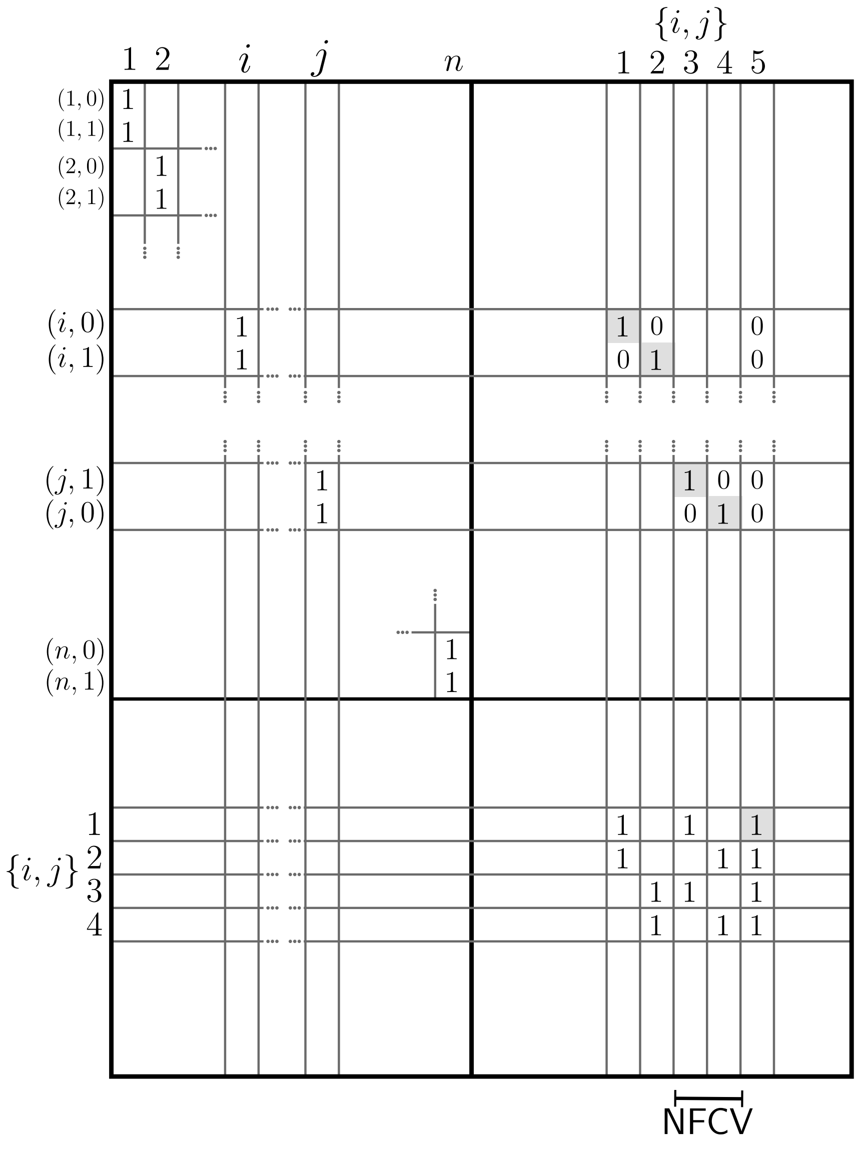
[1] It should be noted that the original definition appearing in Yao's paper [3] requires the players to speak alternatingly. We here adopt the more modern, perhaps one could say *correct*, definition, which uses protocol trees.

ISSN 1433-8092

and $n + 5|E|$ columns, indexed by

$$[n] \cup (E \times [5]).$$

The communication matrix is given by the following picture. All missing entries are 0, and some zeroes are included for emphasis.



We now show that $\chi_1(f_G) = n + 4|E| + \kappa(G)$. Let us start with the "$\leq$" direction. Given a vertex cover $C \subseteq [n]$ of size $k = |C|$, we construct the following 1-partition. ' For each $i \notin C$, we include the single rectangle containing all the ones in the i column. If $i \in C$, we instead include two rectangles containing all the ones in the $(i, 0)$ row and $(i, 1)$ row respectively. Observe that, by construction, these rectangles will cover all the 1s contained in the "upper left" rectangle

$$([n] \times \{0, 1\}) \times [n].$$

Moreover, because C is a vertex cover, for every edge $\{i,j\}$, we must have already covered all the 1s in the $(v,0)$ and $(v,1)$ rows for some $v \in \{i,j\}$. It is then easy to see that we can use only 4 rectangles to cover all the remaining 1s in the submatrix corresponding to that edge.[2] This yields a 1-partition with $n-k+2k+4|E| = n+4|E|+k$ rectangles.

Next we show the "$\geq$" direction. Fix a minimum-sized partition of the 1s of f_G into 1-monochromatic rectangles. We say a rectangle in the partition is a *node rectangle* if it covers $((i,b),i)$ for some $i \in [n]$ and $b \in \{0,1\}$. We form a vertex set $C' \subseteq [n]$ by including i in C' if and only if the entries $((i,0),i)$ and $((i,1),i)$ are covered by different node rectangles in the partition. Observe that we must have at least $n+|C'|$ node rectangles.

We first show that we may assume, without loss of generality, that the given smallest 1-partition has the following property: if $i \notin C'$, then there do not exist two edges $\{i,j\}$ and $\{i,j'\}$ such that the 1 entries in the $(i,0)$ and $(i,1)$ rows of their gadgets share a rectangle. Indeed, suppose we have that three distinct indices i,j,j' such that $\{i,j\}$ and $\{i,j'\}$ are edges of G, $((i,0),(\{i,j\},1))$ and $((i,0),(\{i,j'\},1))$ are in the same rectangle R, and yet $i \notin C'$, meaning $((i,0),i)$ and $((i,1),i)$ are both in the same rectangle S. Then the rectangle R, being 1-monochromatic, cannot include any rows other than $(i,0)$. We may then remove the entry $((i,0),i)$ from the rectangle S it shares with $((i,1),i)$, and put it in R instead. The new 1-partition has the same size, but now $i \in C'$ is forced to hold. So we may assume without loss of generality that we are given a minimum-size 1-partition, where additionally there are no such i,j,j'.

Now, there are two kinds of edges $\{i,j\}$: edges in E_1 are such that $i \notin C'$ and $j \notin C'$, and edges in $E_2 = E \setminus E_1$ are such that $i \in C'$ or $j \in C'$ (or both). Edges in E_1 cannot use the node rectangles to cover the 1s in their gadget, and hence they require at least 5 rectangles. This follows by noting that the five 1s outlined in the figure above form a fooling set. Edges in E_2 of the second kind can use the node rectangles to cover the 1s in the top part of their edge gadget, but they still need 4 rectangles to cover the bottom part. All of the rectangles used to cover the top part must be disjoint, by our without-loss-of-generality assumption above. It follows that $\chi_1(f_G) \geq n+|C'|+5|E_1|+4|E_2| = n+|C'|+4|E|+|E_1|$. By adding one node to C' per edge in E_1 we obtain a vertex cover of size $|C'|+|E_1|$ and so $\chi_1(f_G) \geq n+4|E|+\kappa(G)$, as required.

NP-hardness of $\mathsf{CC}(f)$

We now show how NP-hardness of $\mathsf{CC}(f)$ follows from the above, with a few tricks.

4. We now observe that, for such G, f_G can be computed by a *non-binary* protocol π which is, informally speaking, *fairly balanced.* Indeed, let us define the *binary depth* of a leaf ℓ in a non-binary protocol tree as the sum, over every node in the path from the root to ℓ, of the ceiling of the logarithm of the number of children of the node. The *binary depth* of a non-binary protocol, then, is the maximum binary depth among its leaves. Then, the binary depth of our non-binary protocol for f_G will be at most:

$$\lceil \log(n+2|E|+\kappa(G)) \rceil + 2.$$

We are justified in calling this a *fairly balanced* protocol, since $\chi_1(f_G) = n+4|E|+\kappa(G)$ is a lower-bound on the number of its leaves.

[2] Namely, the submatrix given by the rectangle $A \times B$ where $A = (\{i,j\} \times \{0,1\}) \cup (\{i,j\} \times [4])$ and $B = \{i,j\} \times [5]$

The protocol proceeds as follows. Let $C \subseteq [n]$ be a vertex cover for G. Bob, the column player, speaks first. He sends Alice a bit indicating whether he has a column in the set

$$\mathsf{NFCV} = \{(\{i,j\},c) \in E \times [5] \mid i < j, \text{ and } (i \in C \wedge c \in \{3,4\}) \vee (i \notin C \wedge c \in \{1,2\}\}.$$

NFCV stands for "Not First Cover Vertex". Indeed, Bob tells Alice whether or not his column is one of the two edge-gadget columns (1 and 2, or 3 and 4) corresponding to a vertex which is not the smallest vertex in C. Meaning, Bob sends a 1 to Alice if he has an edge column $(\{i,j\},c)$ such that $i < j$, $i \in C$ and $c \in \{3,4\}$, or an edge column $(\{i,j\},c)$ such that $i < j$, $i \notin C$ (hence $j \in C$ because C is a vertex cover) and $c \in \{1,2\}$. Otherwise, Bob sends a 0.

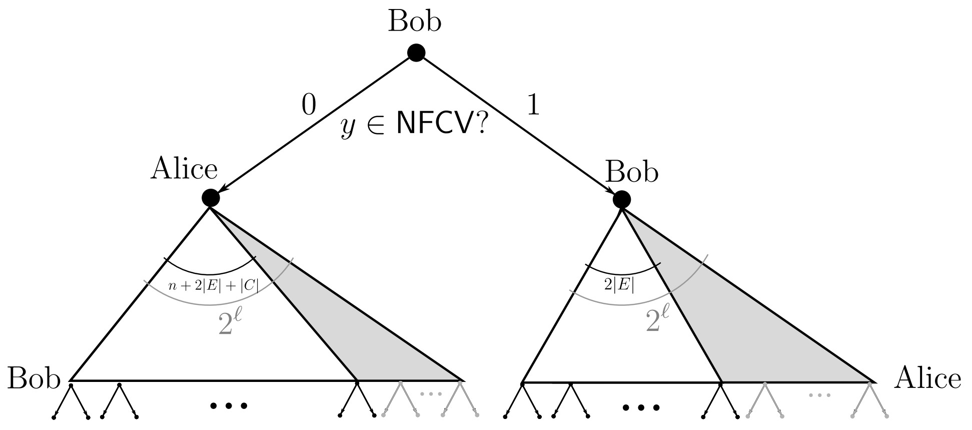
If he sent a 1, meaning he has a column in NFCV, then Bob further tells Alice exactly which column (out of $2|E|$ possibilities) he has, and Alice replies whether Bob's column entry in Alice's row is 0 or 1. This sub-tree of the protocol has exactly $4|E|$ leaves and binary depth $\leq 1 + \lceil \log(2|E|) \rceil + 1 \leq \lceil \log(n + 2|E| + |C|) \rceil + 2$.

Now suppose Bob's column is not in NFCV. Then Alice will tell Bob which of the following cases applies to her row:

(1) She has row (i,r) with $i \in C$, and in this case she sends (i,r) to Bob;
(2) She has a row (i,r) with $i \notin C$, and in this case she sends i to Bob;
(3) She has a row $(\{i,j\},r)$ with $i \in C$ and $r \in \{1,2\}$, or with $i \notin C$ and $r \in \{1,3\}$, and in this case she sends $\{i,j\}$ to Bob;
(4) She has a row $(\{i,j\},r)$ with $i \in C$ and $r \in \{3,4\}$ or with $i \notin C$ and $r \in \{2,4\}$, and in this case she sends $\{i,j\}$ to Bob, also.

After receiving this information, Bob can now tell Alice the value of f_G. Indeed, in case (1), Bob learns exactly which row Alice has. In case (2), Bob learns that Alice has one of two rows $(i,0)$ or $(i,1)$; but $i \notin C$, and Bob's column is not in NFCV, and in the remaining columns, f_G is constant in both of Alice's rows. In case (3), Bob learns, for example, Alice's row is either $(\{i,j\},1)$ or $(\{i,j\},2)$ for some $i \in C$, and one can see from the figure that for the columns outside of NFCV, f_G is constant in both of Alice's rows. Case (3) with $i \notin C$ and both subcases of case (4) are similar.

This sub-tree of the protocol has exactly $2n + 4|E| + 2|C|$ leaves: $4|C|$ for case (1), $2(n - |C|)$ for case (2), $2|E|$ for each of cases (3) and (4). The binary depth of the leaves in this sub-tree is $1 + \lceil \log(n + 2|E| + |C|) \rceil + 1$. This is illustrated in the following figure (we ignore the gray part, for now).



5. Now we do a trick to make the protocol binary and balanced at the same time, where we may formally call a binary tree *balanced* if its depth equals the ceiling of the log of the number of leaves. First, notice the following. Let us convert the above protocol into binary, by encoding each message using $\lceil \log(\text{\# of children}) \rceil$ bits at each node. In this binary protocol, Bob communicates the first bit, and then the two resulting sub-trees are balanced, with the left sub-tree having $2n + 4|E| + 2\kappa(G)$ *useful* leaves, and the right sub-tree having $4|E|$ *useful* leaves, plus some extra *unused* leaves that are left dangling because of our conversion of the protocol into binary — these unused leaves will never be reached by any input to the protocol. Precisely half of the useful leaves are 1-monochromatic, and half are 0-monochromatic.

Now suppose we are given a number k, and we wish to distinguish whether $\kappa(G) \leq k$ or $\kappa(G) > k$. Then let ℓ be the natural number such that $2^{\ell-1} < n + 2|E| + k \leq 2^{\ell}$. Let $d_0 = 2^{\ell} - n - 2|E| - k$, $d_1 = 2^{\ell} - 2|E|$, and $d = d_0 + d_1$. In other words, for the case when $\kappa(G) = k$, d_0 is the number of leaves that are missing from (the useful part of) the left sub-tree of the above nearly balanced protocol, so that it would have *exactly* 2^{ℓ} leaves in this left sub-tree. And d_1 is the number of leaves missing from (the useful part of) the right sub-tree, so that it would *also* have 2^{ℓ} leaves.

Now consider the communication matrix f'_G, given by

$$f'_G = \begin{pmatrix} f_G & 0 \\ 0 & \mathsf{Id}_d \end{pmatrix}$$

where Id_d is the $d \times d$ identity matrix. It now follows that $\chi_1(f'_G) = \chi_1(f_G) + d$, and so

$$\chi_1(f'_G) = 2^{\ell+1} + \kappa(G) - k.$$

On the other hand, a *binary* protocol for f'_G can proceed very similarly to a protocol for f_G. Bob begins by saying whether he has a column in NFCV, or one of the last d_1-many extra columns. In that case, he says which, and Alice replies with the function value. Otherwise, Alice sends a message as before, with an extra case (5) if she has one of the first d_0-many extra rows, and in this case she says which row. Bob then replies with the function value as before. The difference between the old and new protocol is the gray part illustrated in the figure. This protocol has depth exactly $\ell + 2$ if $k \leq \kappa(G)$. And this protocol needs depth $\ell + 3$ if $k > \kappa(G)$. All we are missing is a matching lower-bound, saying that there is *no* protocol that can solve f'_G in depth less than $\ell + 3$, when $k > \kappa(G)$.

6. Now observe that, for every non-constant communication matrix f,

$$\mathsf{CC}(f) \geq \lceil \log \chi_1(f) \rceil + 1.$$

The observation is trivial without the +1, but it is not immediately obvious with it.

Let us prove this by strong induction on $\mathsf{CC}(f)$. Because f is non-constant, we have that $\mathsf{CC}(f) \geq 1$. Our base case is $\mathsf{CC}(f) = 1$. In this case, it is easy to see that $\chi_1(f) = 1$ and thus the theorem holds.

Now suppose that $\mathsf{CC}(f) = s > 1$. Take any optimal protocol for f, and look at its left and right sub-trees. The left sub-tree computes some function f_0 and the right sub-tree computes some function f_1 over disjoint rectangles. Since $\mathsf{CC}(f) > 1$, we have that at least one of f_0 and f_1 is non-constant. Finally, we can conclude that

$$\begin{aligned} \mathsf{CC}(f) &= 1 + \max\{\mathsf{CC}(f_0), \mathsf{CC}(f_1)\} \\ &\geq 2 + \max\{\lceil \log \chi_1(f_0) \rceil, \lceil \log \chi_1(f_1) \rceil\} \\ &\geq 1 + \lceil \log(2 \cdot \max\{\chi_1(f_0), \chi_1(f_1)\}) \rceil \\ &\geq 1 + \lceil \log(\chi_1(f_0) + \chi_1(f_1)\}) \rceil \\ &\geq 1 + \lceil \log \chi_1(f) \rceil, \end{aligned}$$

where the first line uses that the protocol for f is optimal and the second line uses the inductive hypothesis and that at least one of f_0 and f_1 is non-constant (and the fact that $\mathsf{CC}(f) \geq 1 + \lceil \log \chi_1(g) \rceil$ for all non-constant functions f and all constant functions g).

7. Finally, we may conclude the following:

- if $k \leq \kappa(G)$, then $\mathsf{CC}(f'_G) \leq \ell + 2$ by the protocol of §4 and §5, and $\mathsf{CC}(f'_G) \geq \lceil \log(\chi_1(f'_G)) \rceil + 1 = \ell + 2$ by §5 and §6, so $\mathsf{CC}(f'_G) = \ell + 2$.
- if $k > \kappa(G)$, however, we get $\mathsf{CC}(f'_G) \geq \lceil \log(\chi_1(f'_G)) \rceil + 1 = \ell + 3$.

And so it follows that communication complexity $\mathsf{CC}(f)$ of a given function f is NP-hard to compute exactly.

8 *Final remarks.* It should be noted that the above result does not give us *any* hardness-of-approximation. It could well be that $\mathsf{CC}(f)$ is *not* NP-hard to compute for an additive error of 1! Do recall, however, that Kushilevitz and Weinreb have shown that approximating $\mathsf{CC}(f)$ up to a factor of (roughly) 1.1 is *cryptographically hard* [2], so we do expect that the problem is *hard* to approximate, just not necessarily *NP-hard* to approximate.

As we mentioned in footnote, we have proven NP-hardness of communication complexity, as defined by the smallest depth of a protocol tree, and not as per Yao's original definition, of the smallest number of rounds in an alternating protocol. These two definitions of communication complexity are the same up to a constant factor of 2, but our proof is not robust up to such a factor, hence, strictly speaking, Yao's original question remains unanswered. We are currently working on this problem: it seems to require significantly new ideas.

We should also remark that the reduction of Jiang and Ravikumar, together with the known hardness of approximation results for vertex cover, already show that $\chi_1(f)$ is hard to approximate up to some constant $\gamma > 1$. One would think that this would lead to a similar hardness-of-approximation for $L(f)$, the smallest number of leaves in a protocol for f, but it is unclear how to generalize the crucial observation of §6. The analogous conjecture to §6 for L, which would state that $L(f) \geq 2\chi_1(f)$, is not true: certain functions with imbalanced protocols serve as a counter-example.

As such our result, which does hold for the usual model of communication complexity, is not very robust. What one would ideally like is an NP-hardness of approximation result for $\mathsf{CC}(f)$, up to a reasonably large constant factor (e.g. 4 would be enough, but not 2), since this would make the hardness robust up to rebalancing the protocol tree. This would require reducing from a different problem, such as graph coloring, since vertex cover can be approximated with a factor of 2, and so it cannot prove *any* hardness of approximation on $\mathsf{CC}(f)$.

In a very distant future one could even hope for hardness of approximation up to a small exponent. Proving hardness of approximation to an arbitrary constant exponent would disprove the log rank conjecture, assuming $\mathsf{P} \neq \mathsf{NP}$, and we have even wondered if this was a viable way to attack the log-rank conjecture.

In summary: the question of whether $\mathsf{CC}(f)$ is hard to approximate is wide open, and an entirely new approach will have to be developed to attack this question.

References

1 Tao Jiang and Bala Ravikumar. Minimal NFA problems are hard. *SIAM Journal on Computing*, 22(6):1117–1141, 1993.

2 Eyal Kushilevitz and Enav Weinreb. On the complexity of communication complexity. In *Symposium on Theory of Computing* (STOC), pages 465–474, 2009.

3 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing. In *Symposium on Theory of Computing* (STOC), page 209–213, 1979.