Electronic Colloquium on Computational Complexity, Report No. 75 (2003)

A Tutorial on the Deterministic two-party Communication Complexity

Agostino Capponi

Thales Nederland B.V.[*] University of Twente (EEMCS) [†]

September 7, 2003

Abstract

Communication complexity is concerned with the question: how much information do the participants of a communication system need to exchange in order to perform certain tasks? The minimum number of bits that must be communicated is the deterministic communication complexity of f. This complexity measure was introduced by Yao [1] to focus attention on the cost of information transfer associated with a given distributed computation. Subsequently the model has been studied for its own sake as an interesting abstract model of computation. Here we present a tutorial on the two-party deterministic communication complexity. First we illustrate the basic algebraic and combinatorial tools used in this theory. Then we discuss the communication complexity of specific functions and some of the many applications that the theory of communication complexity has. We also dedicate one paragraph to briefly mention some variants of the basic model of the deterministic two-party communication complexity.

1 Introduction

Consider a two argument Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$. Let Alice and Bob be the two communicating parties. Alice is given an input $x \in \{0,1\}^n$ and Bob is given an input string $y \in \{0,1\}^n$. Both know the definition

[*]Thales Nederland B.V., Zuidelijke havenweg 40, PO box 42 - 7550 GD Hengelo(Ov), The Nederlands, e-mail: agostino.capponi@nl.thalesgroup.com

[†]Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS), University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherland

ISSN 1433-8092

of the function f, but each of them does not know the input of the other. They wish to compute the value of $f(x, y)$ cooperatively. The computation of the value $f(x, y)$ is done using a **communication protocol**. During the execution of the protocol, the two parties alternate roles in sending messages. Each message is a string of bits. The protocol specifies if the execution has terminated (in this case it also specifies what is the output). If the execution has not terminated , the protocol specifies what message the sender (Alice or Bob) should send next, as a function of its input and of the communication so far. A communication protocol computes the function f, if for every input pair $(x, y) \in \{0,1\}^n \times \{0,1\}^n$ the protocol terminates with the value $f(x, y)$ as its output. It is worth noting that every function can be computed by the following trivial protocol: Alice sends her input x to Bob (n bits of computation) and then Bob who now knows both x and y can compute the bit $f(x, y)$ and then send this bit to Alice. However, if we think of large n, this is very expensive. In many cases much more efficient protocols can be designed and we will show some of them. The study of communication complexity aims at identifying the minimum number of bits that can be used to compute a function f. More precisely the complexity measure for a protocol P is the number of bits exchanged by the two parties in the worst case. Formally, let $s_P(x, y) = (m_1, m_2, \ldots, m_r)$ be the sequence of messages exchanged on input (x, y) during the execution of P, where m_i denotes the i-th message sent in the protocol. From here on we will call each sequence of messages a **conversation**. Then denote by $|m_i|$ the length (number of bits) of m_i and let $|s_P(x, y)| = \sum_{i=1}^{r} |m_i|$. At this point, we define the communication complexity of P as the worst case number of bits exchanged by the protocol. In formal settings we have:

$$D(P) \triangleq \max_{(x,y)\in\{0,1\}^n\times\{0,1\}^n} |s_P(x, y)|.$$

The (deterministic) communication complexity of a function f is the communication complexity of the best protocol that computes f, that is

$$D(f) \triangleq \min_{P:P \text{ computes } f} D(P).$$

As discussed earlier, it is obvious to see that

Proposition 1.1. *For any Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, $D(f) \leq n + 1$.*

Proof. Alice sends all her input to Bob (requiring n bits using an appropriate encoding). Bob computes $f(x, y)$ privately (with his unlimited computational power) and then he sends the answer back to Alice (only one bit). □

The rest of the paper is organized as follows. In the next section we illustrate the most known combinatorial methods used to approximate (in some cases calculate) the deterministic communication complexity of a function. In Section 3 we discuss algebraic methods used to work out bounds for the deterministic communication complexity of a function. In Section 4 we apply the results illustrated in the previous sections to determine the communication complexity of some functions. Section 5 discusses some variants of the basic model of the two-party communication complexity. Finally in the last section we describe some applications of the communication complexity theory, basically to distributed computation, networks and VLSI circuits.

2 Combinatorial approaches

Let us focus our attention on arbitrary functions (not necessarily Boolean functions). It is rather intuitive to see that fixed a Boolean function $f : X \times Y \to Z$ it is not possible to use less than the number of bits needed to represent all the possible values of z that the function can assume. In fact the parties must exchange different sequences of messages to compute different values of z. Formally:

Proposition 2.1. *Let $f : X \times Y \to Z$ be a non Boolean function. Then*

$$D(f) \geq \log_2 |\mathrm{Range}(f)| \text{ where}$$
$$\mathbf{Range(f)} = \{z \in Z : \exists (x,y) \in X \times Y : f(x,y) = z\}.$$

Proof. If $f(x,y) \neq f(x_1,y_1)$, then any communication protocol must use different sequences of messages on (x,y) and (x_1,y_1) to compute the exact value of f. That means that at least $|\mathrm{Range(f)}|$ different conversations are possible, therefore at least $\log_2 |\mathrm{Range(f)}|$ bits are needed to represent them all. □

The inequality above is not very informative when we consider Boolean functions. In this case it simply states that more than one bit must be used to compute functions that are not boolean. The success in proving good lower bounds on the communication complexity of Boolean functions comes from the analysis of the combinatorial structure imposed by protocols. The basic element in the study of the combinatorics of the protocols is a rectangle.

A **rectangle** is a subset of $\{0,1\}^n \times \{0,1\}^n$ of the form $A \times B$, where each of A and B is a subset of $\{0,1\}^n$. It is said to be **f-monochromatic** if, for every $x \in A$ and $y \in B$, the value $f(x,y)$ is the same. If the value is 0, we call it an $(f,0)$-monochromatic rectangle. It it is 1, we call it an $(f,1)$-monochromatic rectangle. It is true that:

Proposition 2.2 ([1]). *Let P be a protocol that computes a function f and $(m_1,\ldots,m_r)$ be a sequence of messages. The set of inputs for which $s_P(x,y) = (m_1,\ldots,m_r)$ is an f-monochromatic rectangle.*

Proof. First, something stronger is proved. By induction on i it is showed that the set of inputs (x,y) for which the conversation starts with $(m_1,\ldots,m_i)$ is a rectangle (we are not requiring that is f-monochromatic).

Basic step. Suppose that m_1 is sent by Alice. Let A be the set of inputs x on which Alice starts the conversation with m_1. Then the rectangle $A \times \{0,1\}^n$ is the set of inputs (x,y) on which the conversation between Alice and Bob starts with m_1. We can repeat a similar reasoning if it is Bob to start the conversation.

Inductive step. Let the rectangle $R = A \times B$ be the set of inputs on which the conversation between Alice and Bob starts with $(m_1, \ldots, m_i)$. At this point m_{i+1} can be sent by Alice or Bob. Suppose that the message m_{i+1} is sent by Alice. Consider the set $A_1 \subset A$ such that given the input x and the sequence of messages $(m_1, \ldots, m_i)$, Alice sends the message m_{i+1}. It is obvious that the rectangle $A_1 \times B$ is the set of inputs on which the conversation starts with $m_1, \ldots, m_{i+1}$. Again we can repeat an analogous reasoning if m_{i+1} is sent by Bob.

It remains now to show that the rectangle is f-monochromatic when the communication ends. This is true because the protocol P computes f, so all the inputs in the rectangle obtained at the end of the conversation must have the same f-value. □

Note that each rectangle corresponds to a different conversation between Alice and Bob.

Given a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, denote by $C^P(f)$ the minimum number of f-monochromatic rectangles needed to partition the space of inputs $\{0,1\}^n \times \{0,1\}^n$. Then it is true that:

Proposition 2.3 ([2])**.** *For every function* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$,

$$D(f) \geq \log_2 C^P(f).$$

Proof. Every protocol P partitions the space of inputs $\{0,1\}^n \times \{0,1\}^n$ into f-monochromatic rectangles by the Proposition 2.2. The number of these rectangles is at most $2^{D(P)}$ and $D(f) \leq D(P)$, therefore $C^P(f) \leq 2^{D(f)}$. □

The proposition 2.3 tells us that in order to prove lower bounds on the communication complexity of f, one can prove lower bounds on the minimum number of f-monochromatic rectangles needed to partition the space of inputs. Note that in some cases, lower bounds only on the number of $(f,0)$-monochromatic or $(f,1)$-monochromatic rectangles allow us to deduce tight lower bounds on the communication complexity of f. We will see this in detail in the fourth section of this paper.
Another interesting method to prove lower bounds on $C^P(f)$ is the **fooling set** method. This method was implicited used in [1] and appeared explicitly in [4]. It is based on exhibiting a large set of inputs such that each of them must be in a different f-monochromatic rectangle. More precisely a set of input pairs $\{(x_1, y_1), (x_2, y_2), \ldots, (x_l, y_l)\}$ is called an (f,z) fooling set ($z \in \{0,1\}$) if:

- For all i, $f(x_i, y_i) = z$.
- For all $i \neq j$, either $f(x_i, y_j) \neq b$ or $f(x_j, y_i) \neq z$.

A very challenging problem is to decide whether or not $D(f) = O(\log_2 C^P(f))$. This question was posed by Lovasz and Saks [7]. So far, it is only known that $D(f) = O(\log_2 C^P(f))^2)$.

The following proposition [2], which we only mention, provides with another tool to prove lower bounds on the communication complexity of f.

Proposition 2.4. *Given an $(f, 0)$ and an $(f, 1)$ fooling set of size l and l_1 respectively, then $D(f) \geq \log_2(l + l_1)$.*

We now discuss methods used to prove upper bounds on the communication complexity of a function. Before illustrating the results, we give some useful definitions. Let $f : X \times Y \to \{0, 1\}$. The **protocol partition number**, denoted by $C^L(f)$, is the minimum number of different conversations that Alice and Bob must have when f is the given function. The **cover number** of f, $C(f)$, is the minimum number of f-monochromatic rectangles needed to cover $X \times Y$ (note that intersections between f-monochromatic rectangles are possible). If $z \in \{0, 1\}$, $C^z(f)$ is the minimum number of f-monochromatic rectangles needed to cover the z-inputs of f (i.e (x, y) such that $f(x, y) = z$). It can be immediately deduced from the definitions that $C(f) \leq C^P(f) \leq C^L(f)$ and that $C(f) = C^0(f) + C^1(f)$. In [2] it is stated:

Proposition 2.5. *For every Boolean function $f : X \times Y \to \{0, 1\}$ and $z \in \{0, 1\}$, $D(f) \leq C^z(f) + 1$.*

Proof. Set k to $|C^z(f)|$. Alice and Bob agree on a family $\mathscr{F} = \{R_1, R_2, \ldots, R_k\}$ of (f, z)-monochromatic rectangles covering all the z-inputs of f. Then they use the following two-phase protocol P to compute f:

- Alice sends to Bob a binary string $p = (a_1, a_2, \ldots, a_k)$ such that for every $i \in \{1, \ldots, k\}$, $a_i = 1$ if $x \in R_i$ and 0 otherwise.
- Bob checks if there exists a rectangle R_i such that $y \in R_i$ and $a_i = 1$. If so, then he sends to Alice the bit z. If such a rectangle does not exist then he sends to Alice the complement of the bit z.

We now prove that the protocol is correct i.e if $P(x, y) = k$, then $f(x, y) = k$. If $k = z$, then there exists a rectangle R_i such that $(x, y) \in R_i$. Since R_i is an (f, z)-monochromatic rectangle, we get that $f(x, y) = z$. If $k = \bar{z}$ (complement of the bit z), there is not any rectangle R_i such that $(x, y) \in R_i$. Thus (x, y) is not a z-input of f, i.e $f(x, y) = \bar{z}$. □

The proposition just proved allows us to assert that:

$$\log_2 (C^z(f)) \leq \log_2 (C^P(f)) \leq D(f) \leq C^z(f) + 1.$$

It is possible that a certain protocol induces a small number of f-monochromatic rectangles to partition its input space, yet requires the players to exchange a large number of bits (in the worst case). In other words it has a small number of conversations, but the longest one requires much more bits than the logarithm of the number of all the possible conversations. J.Sgall [8] noted that $D(f) \leq 3\log_2 C^L(f)$. Before discussing it, we observe that a protocol for f can be seen as a labeled binary tree where:

- The label of an edge is 0 if it connects a vertex with his left son and 1 if it connects a vertex with his right son.
- Each conversation $(m_1, m_2, \ldots, m_r)$ to compute $f(x, y)$ (note that m_i is a sequence of bits) corresponds to a path labeled $m_1 m_2 \ldots m_r$ of the tree. The last vertex of the path is a leaf associated with the value $f(x, y)$.

Note that there is one to one correspondence between conversations and leaves of a protocol tree for f.

Proposition 2.6. *For every Boolean function f, $D(f) \leq 3\log_2 C^L(f)$.*

Proof. The basic idea is that a protocol tree for f with a certain number of leaves can be converted into a balanced protocol in which the depth is about the logarithm of the number of leaves. Before describing the protocol, we show:

Combinatorial Fact. Let T be a binary tree with t leaves. For every node w of the tree denote by t_w the number of leaves of the subtree rooted in w. If $t \neq 1$, there exists a node v of T such that $t_v > t/2$, $t_{left(v)} \leq t/2$ and $t_{right(v)} \leq t/2$, where $left(v)$ and $right(v)$ denote respectively the left and the right son of v.

Verify. We design an algorithm to detect a node that verifies the property given above. The recursive algorithm *Find*(x, t) goes as follows:

- If $t_{left(x)} > t/2$ then *Find*$(left(x), t)$;
- If $t_{right(x)} > t/2$ then *Find*$(right(x), t)$;
- If $t_{left(x)} \leq t/2$ and $t_{right(x)} \leq t/2$ then return x

The input of the algorithm is the root of T. The algorithm always stops because T contains a finite number of nodes. Note that the algorithm cannot return a leaf. If by absurd it should return a leaf l, then it would mean that $t_l > t/2$ (just look at how the algorithm works). But $t_l = 1$ because l is a leaf of the tree. Hence the previous inequality can be true only if $t = 1$, but that contradicts the hypothesis of the combinatorial fact. The node v returned from the algorithm is such that $t_v > t/2$, $t_{left(v)} \leq t/2$ and $t_{right(v)} \leq t/2$. We only need to show that both $t_{left(v)}$ and $t_{right(v)}$ are greater than 0 and therefore they both exist (t_v is different from 0 for any vertex v). But $t_{left(v)} + t_{right(v)} = t_v$ for any node v which is not a leaf of the tree and this is just our case. Since both $t_{left(v)}$ and $t_{right(v)}$ are not greater than $t/2$ the equality can be verified only when both $t_{left(v)}$ and $t_{right(v)}$ are greater than 0. ∎

The protocol P makes its computation on a tree protocol T for f with $C^L(f)$ leaves. In order to make clearer the protocol P, we introduce some symbols: given a protocol tree for f with $C^L(f)$ leaves and a vertex v of the tree, R_v denotes the set of all input pairs (x, y) such that the computation of $f(x, y)$ passes through the node v of T. We call z an X-**representant** in R_v if $(z, y) \in R_v$ for some $y \in \{0,1\}^n$. We will say that z is an Y-**representant** in R_v if $(x, z) \in R_v$ for some $x \in \{0,1\}^n$. As follows, we present the protocol $P(x, y)$.

- Alice and Bob run the algorithm $Find(root(T), C^L(f))$ ($root(T)$ denotes the root of T). If the algorithm does not return any node, the combinatorial fact tells us that $t = 1$, i.e T consists of a single leaf node. Alice and Bob consider the value associated with that leaf as their output. If the algorithm returns a node v, then Alice sends to Bob a string of two bits (i, j) defined as follows. The bit i is set to 1 if x is an X-representant of $R_{s(v)}$ and 0 otherwise. The bit j is set to 1 if x is an X-representant of $R_{d(v)}$ and 0 otherwise.

- Bob sends 1 either if $i = 1$ and $y \in R_{s(v)}$ or if $j = 1$ and $y \in R_{d(v)}$. He sends 0 either if $i = 1$ and $y \notin R_{s(v)}$ or if $j = 1$ and $y \notin R_{d(v)}$. Bob does not send any bit if $i = j = 0$.

- If Bob sent 1, they both execute the protocol on $T_{s(v)}$ if $i = 1$ or on $T_{d(v)}$ if $j = 1$ (for a node w, T_w denotes the subtree of T rooted in w). If Bob sent 0 or no bit, then they both execute the protocol on T^1, where T^1 is obtained from T by substituting T_v with a leaf associated with the value 0.

It is trivial to see that if l is the leaf of the tree T such that R_l contains the input pair (x, y), then all the new trees on which the protocol P is executed contain the leaf l associated with the same value $f(x, y)$. Alice and Bob consider as their output the value associated with the leaf of the tree on which the protocol is executed for the last time. That leaf will certainly be l (just look at how the protocol works and consider that each input pair is associated with only an R_n, where n is a leaf of the tree). Since the value associated with the leaf l has never been modified in the previous iterations, it is still associated with the value $f(x, y)$. Hence the protocol is correct.

It remains now to calculate the computational cost of the protocol. Let t be the number of the leaves of the tree on which the protocol is executed in a generic iteration and v the node returned from the algorithm $Find$ in the same iteration. In the next iteration the protocol P can be executed on:

- $T_{s(v)}$. In that case $t_{s(v)} \leq t/2$.

- $T_{d(v)}$. In that case $t_{d(v)} \leq t/2$.

- T^1 obtained by substituting T_v with a leaf. Then T^1 has $t - t_v + 1$ leaves. But $t - t_v + 1 \leq t - t/2 + 1 \leq t/2$ because any vertex v returned from the algorithm $Find$ is such that $t_v \geq t/2$.

Considering also that at each iteration Alice sends to Bob two bits and Bob replies by sending at most one bit. We can deduce that the worst case number of bits exchanged by both parties can be described by the following recursive relation $L(1) = 0, L(t) \leq 3 + L(t/2)$. It is satisfied when $L(t) \leq 3\log_2 t$. Since the protocol starts its computation on a tree with $C^L(f)$ leaves, we can conclude that the communication complexity of f is $D(f) \leq 3\log_2 C^L(f)$. □

3 Algebraic approaches

In the previous section we have illustrated methods based upon combinatorial theory. In this section we show methods of algebraic nature that have been used to provide bounds on the communication complexity of boolean functions. Before discussing these methods we illustrate how the problem of the deterministic communication complexity of boolean functions can be formalized in an algebraic way in terms of matrices. In fact we can associate with every function $f : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$ a matrix M_f of dimensions $|X| \times |Y|$. The rows of M_f are indexed increasingly by the elements of X (the index of the first row is $x_1 = 0^n$, that of the second row is $x_2 = 0^{n-1}1$ and so on) and so are the columns by the elements of Y. The (x, y) entry of M_f is simply defined as $f(x, y)$. We define **rank**(f) as the linear rank of the matrix M_f over the field of reals. The argument of the rank is very useful to prove lower and upper bounds on the communication complexity of Boolean functions. The rank lower bound shown in the following proposition was presented in [3].

Proposition 3.1. *For any Boolean function f, $D(f) \geq \log_2$ (2rank$(f) - 1$).*

Increasing gaps between this lower bound and the communication complexity were demonstrated in [10], [11] and [12]. Another interesing gap is due to Nisan and Widgerson [13]. They exhibit a function f such that
log (2rank(f)-1) is $O(n^{0.631})$, while the communication complexity of f is $\Omega(n)$.

Let us consider now upper bounds to the communication complexity of boolean functions. The best known is illustrated in the next proposition.

Proposition 3.2. *Let $f : X \times Y \rightarrow \{0,1\}$ be a Boolean function. Then $D(f) \leq$ rank$(f) + 1$.*

Proof. Consider the following matrix:

$$M_f = \begin{pmatrix} z_{11} & z_{12} & \ldots z_{12^n} \\ z_{21} & z_{22} & \ldots z_{22^n} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ z_{2^n1} & z_{2^n2} & \ldots z_{2^n2^n} \end{pmatrix}$$

where $z_{x,y} = f(x, y)$

We can always find a $2^n * p$ submatrix $M^1(f)$ of M_f, $p = \text{rank}(f)$, where the p column vectors of M_f^1 constitute a base of the column vectors of M_f.

$$M_f^1 = \begin{pmatrix} z_{1j_1} & z_{1j_2} & \dots z_{1j_p} \\ z_{2j_1} & z_{2j_2} & \dots z_{2j_p} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ z_{2^n j_1} & z_{2^n j_2} & \dots z_{2^n j_p} \end{pmatrix}$$

Note that, for every column vector z of M_f, there exists a vector $a = (a_1, a_2, \dots, a_p)$ of scalars such that matrix product of M_f^1 and a is z.
We can always extract a square submatrix M_f^2 of M_f^1 whose dimension is p, where both the set of rows and of columns constitute a linearly independent set. This must be true because the row rank and the column rank of a matrix are the same.

$$M_f^2 = \begin{pmatrix} z_{i_1 j_1} & z_{i_1 j_2} & \dots z_{i_1 j_p} \\ z_{i_2 j_1} & z_{i_2 j_2} & \dots z_{i_2 j_p} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ z_{i_p j_1} & z_{i_p j_2} & \dots z_{i_p j_p} \end{pmatrix}$$

Note that, for any string $z \in \{0,1\}^p$, there exists a vector $a = (a_1, a_2, \dots, a_p)$ of scalars such that matrix product of M_f^2 and a is z. Moreover, such a vector a is unique because M_f^2 is a $p * p$ matrix and its rank is p.
At this point, we are ready to design the protocol P for f. It consists of two stages:

- Bob sends to Alice the string $z_{i_1 y}, z_{i_2 y}, \dots, z_{i_p y}$.
- Alice calculates the vector a of p scalars such that the matrix product of M_f^2 and a is the vector $(z_{i_1 y}, z_{i_2 y}, \dots, z_{i_p y})$. Then he calculates the full column vector $z_{1y}, z_{2y}, \dots, z_{2^n y}$ as matrix product of M_f^1 and the vector a. It is not possible that $z_{1y}, z_{2y}, \dots, z_{2^n y}$ is given by the product of M_f^1 and a vector b different from a. If it were so, then the product of M_f^2 and b would be the vector of p components $(z_{i_1 y}, z_{i_2 y}, \dots, z_{i_p y})$. That contradicts the fact that, for any string $z \in \{0,1\}^p$, there is only one vector v such that the matrix product of M_f^2 and v is the vector z. Finally Alice sends to Bob the $z_{x,y}$ entry of M_f.

It is quite obvious that this protocol uses p bits (sent by Alice) plus 1 bit (sent by Bob) and it computes the function correctly. Since p is the rank of f the proof is complete.

$\square$

Another interesting bound for the commmunication complexity of Boolean functions is obtained using the argument of the rank combined with the one of monochromatic rectangles. This approach is evident in the following

Proposition 3.3 ([2]). *Let $f : X \times Y \to \{0,1\}$ a function. Then $D(f) = O(\log_2 C^0(f)\ \log_2 \mathrm{rank}(f))$*

Proof. Let C be a minimum covering of the 0-inputs of f with $(f,0)$-monochromatic rectangles. Clearly, $|C| = C^0(f)$. Let A be the largest $(f,0)$-monochromatic rectangle of M_f in C. Then A induces in a natural way a partition of M into 4 submatrices A, B, C, D with B sharing the rows of A and C sharing the columns of A. Clearly, $\mathrm{rank}(B) + \mathrm{rank}(C) \leq \mathrm{rank}(M_f) + 1$. Assume that $\mathrm{rank}(B) \leq \mathrm{rank}(C)$. That implies that the submatrix $A|B$ has rank at most $2 + \mathrm{rank}(f)/2$. The protocol used by Alice and Bob is the following. Alice sends to Bob a bit saying whether or not his input x belongs to the rows of A. The players then continue recursively with a protocol for the submatrix $A|B$ if the input y of Bob belongs to the columns of A, or for the submatrix $C|D$ if it does not. If $\mathrm{rank}(B) \leq \mathrm{rank}(C)$, it only needs to change the protocol in this way: Bob sends his input saying whether or not his input y belongs to the columns of A. The players then continue recursively with a protocol for the submatrix $A|C$, or for the submatrix $B|D$, according to the bit communicated. If we carefully examine the protocol, we see that at each step Alice and Bob exchange one bit. Then they both continue the execution of the protocol either on a matrix whose rank is less than the previous rank plus two or on a matrix where one $(f,0)$ rectangle has been eliminated. Therefore the number of different conversations is described by the following recurrence relation $L(r,m) \leq L(r/2+2,m) + L(r,m-1)$ starting with $r = \mathrm{rank}(f)$ and $m = |C^0(f)|$ (note that r must necessarily be different from 0). The basic steps of the recurrence are $L(1,m) = L(r,1) = 1$. By induction on k and l, we get that $L(r,m) \leq (m+1)^{\log(r)}$. By substituting r and m respectively with $\mathrm{rank}(f)$ and $C^0(f)$, we obtain that the number of conversations is $L(\mathrm{rank}(f), C^0(f)) \leq (C^0(f)+1)^{\log_2 \mathrm{rank}(f)}$. Note that each conversation is coded by a different sequence of bits. Hence, we can assert that the number of bits used by Alice and Bob with this protocol is $\log_2(C^0(f) + 1)^{\log_2 \mathrm{rank}(f)}$. Since $D(f)$ is the minimum number of bits used by any protocol, we get that $D(f) = O(\log_2 C^0(f)\ \log_2 \mathrm{rank}(f))$. ☐

Similarly it is possible to prove that $D(f) = O(\log_2 C^1(f)\ \log_2 \mathrm{rank}(f)))$. A very challenging open problem has been posed in [7] and asks:

Open Poblem: Does $D(f) = \log_2(\mathrm{rank}(f))^{O(1)}$, for all boolean functions f?

Any improvement of either the lower bound for the gap 3.1 or the upper bound 3.2 is very interesting.

4 Communication complexity of particular functions

In this section we apply the combinatorial and algebraic approaches described in the two previous sections to determine the communication complexity of specified functions. The first two functions that we examine, Eq and Gt, were first defined and discussed by Yao in the seminal paper [1].

Let us start with the equality function $Eq : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$:

$$Eq(x,y) \triangleq \begin{cases} 1 & \text{if x = y} \\ 0 & \text{otherwise} \end{cases}$$

An $(Eq, 1)$ fooling set of size 2^n is $\{(\alpha,\alpha) : \alpha \in \{0,1\}^n\}$ (because for every α, $Eq(\alpha,\alpha) = 1$), whereas for every $\alpha \neq \beta$, $Eq(\alpha,\beta) = 0$. It follows from Proposition 2.4 that $D(Eq) \geq n$.

A tighter bound is obtained considering the rank of Eq. Since M_{Eq} is the identity matrix its rank is 2^n. If we apply the proposition 3.1, we obtain that $D(Eq) \geq \log_2(2^{n+1} - 1)$ i.e $D(Eq) \geq n + 1$. Since it is trivially true that $D(Eq) \leq n + 1$ (proposition 1.1), we can assert that $D(Eq) = n + 1$.

Consider now the function $Gt : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ so defined:

$$Gt(x,y) \triangleq \begin{cases} 1 & \text{if } x > y \\ 0 & \text{otherwise} \end{cases}$$

We will determine $D(Gt)$ both using a combinatorial approach and an algebraic approach.

Proposition 4.1. *(combinatorial proof).* $D(Gt) = n + 1$.

Proof. We state that $S_0 = \{(x,y) : x = y\}$ and $S_1 = \{(x,y) : x - y = 1\}$ are respectively a $(Gt, 0)$ and a $(Gt, 1)$ fooling set. As regards to S_0, we show as follows that two necessary conditions for it to be a fooling set are verified:

- $\forall (x,y) \in S_0$, $Gt(x,y) = 0$ because $x = y$.
- If (x,y) and (x_1,y_1) are in S_0, suppose w.l.o.g that $x \leq x_1$. Then $x_1 > y$, hence $Gt(x_1,y) = 1$.

It is easy to see that $|S_0| = 2^n$.

Let us see now that also S_1 verifies the two conditions to be a fooling set. In fact:

- $\forall (x,y) \in S_1$, $Gt(x,y) = 1$ because $x = y + 1$, hence $x > y$.
- If (x,y) and (x_1,y_1) are in S_1, suppose w.l.o.g that $x \geq x_1$. Then $x_1 \leq y$, hence $Gt(x_1,y) = 1$

It is easy to see that $|S_1| = 2^n - 1$.

At this point, if we apply the proposition 2.4, we obtain that $D(Gt) > n$. Since the proposition 1.1 tells us that $D(Gt) \leq n+1$, we get $D(Gt) = n+1$. $\square$

Proposition 4.2. *(algebraic proof).* $D(Gt) = n + 1$.

Proof. Consider the matrix M_{Gt}.

$$M_{Gt} = \begin{pmatrix} 0 & 0 & 0 & \dots 0 \\ 1 & 0 & 0 & \dots 0 \\ 1 & 1 & 0 & \dots 0 \\ 1 & 1 & 1 & \dots 0 \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & \dots 1 \end{pmatrix}$$

Note that this matrix has 2^n rows. If we consider a set containing all the rows of the matrix except the first one, we immediately see that this set is linearly independent. Therefore, we can conclude that $\text{rank}(Gt) = 2^n - 1$. Applying now the proposition 3.1 we obtain that $D(Gt) > n$. Finally applying the trivial proposition 1.1, we get $D(Gt) = n + 1$. $\square$

Babai, Frank and Simon [23] considered the **inner product** function $Ip : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ defined by

$$Ip(x,y) = \sum_{i=1}^{n} \pi_{(x,i)} \pi_{(y,i)} \bmod 2,$$

where $\pi_{(z,i)}$ denotes the projection of z with respect to its ith coordinate (i.e the ith coordinate of z). It is known that:

Proposition 4.3. *Any $(Ip, 0)$-monochromatic rectangle covers at most 2^n of the input pairs.*

This proposition implies [2]:

Proposition 4.4. $D(Disj) = \Omega(n)$, *where* $Disj : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *is defined by*

$$Disj(x,y) \triangleq \begin{cases} 0 & \textit{if there exists } i\textit{: } \pi_{(x,i)} = \pi_{(y,i)} \\ 1 & \textit{otherwise} \end{cases}$$

Proof. We begin by noting that if D is a $(Disj, 1)$-monochromatic rectangle, then D is also an $(Ip, 0)$-monochromatic rectangle. This is true because if $(x,y) \in D$, then by definition of $Disj$, $\pi_{(x,i)} \neq \pi_{(y,i)}$ for every $i \in \{1, \dots, n\}$. Hence, by definition of Ip, $Ip(x,y) = 0$. By proposition 4.3, we know that any $(Ip, 0)$-monochromatic rectangle covers at most 2^n of the input pairs. From this, we can deduce that the size of any $(Disj, 1)$-monochromatic rectangle is at most

2^n. Now, let us count all the 1-inputs of $Disj$ with the following method: for every i, consider all the strings having exactly a number of ones equal to i. For any such string z, we count how many strings have zero's in the positions where z has ones. In this way, we get that there are $\sum_{i=0}^{n} \binom{n}{i} 2^{n-i} = 3^n$ ones. Since the size of any $(Disj, 1)$-monochromatic rectangle is at most 2^n, we get that at least $(3/2)^n$ $(Disj, 1)$-monochromatic rectangles are needed to partition the 1-input pairs of $Disj$. At this point, we apply the proposition 2.3 and obtain that $D(f) \geq \log_2 (3/2)^n = \Omega(n)$. □

The functions studied so far attain the worst communication complexity for a Boolean function. We will exhibit now a function whose communication complexity is better. This is the function $Max : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ [2], where both x and y are respectively the characteristic vectors of a set $X \subseteq \{1, \ldots, n\}$ and a set $Y \subseteq \{1, \ldots, n\}$:

$$Max(x, y) \triangleq \begin{cases} 1 & \text{the maximum in } X \text{ is not smaller than the maximum in } Y \\ 0 & \text{otherwise} \end{cases}$$

A simple protocol to compute $Max(x, y)$ is the following: Alice sends to Bob the maximum number in x ($\log_2 n$ bits are needed to code it). Bob compares this number with the maximum number in y. If his number is greater then he sends to Alice the output 0, otherwise he sends to Alice the output 1. Thus we have that $D(Max) \leq \log_2 n + 1$.

5 Variants of the two party model

In this section we briefly describe some variants of the two-party model. For each variant we give a short description and pointers to the literature.

Non deterministic protocols. Lipton and Sedgewick [4] defined the non deterministic protocols. They allow the two parties to choose the messages that they send in a non deterministic way. The only important thing is that there is an execution of the protocol that leads to the correct answer. The combinatorial objects used are coverings, i.e we relax the need to partition the space $X \times Y$ into f monochromatic rectangles by allowing intersections between monochromatic rectangles. A fundamental result due to Aho et Al. [14] relates the deterministic communication complexity of a function to its non deterministic communication complexity.

Rounds. In the definition of communication complexity we are interested only in the number of bits exchanged between Alice and Bob and we ignore the number of rounds, i.e the number of messages exchanged. We may ask how many rounds are really necessary to obtain a low cost communication protocol. For instance, maybe it always suffices for Alice to send one message (containing several bits) to Bob and then Bob can compute the answer by himself. Such a protocol is said to be a one-round protocol. The one round communication complexity of a function, i.e the minimum number of bits exchanged by Alice and Bob when only one round is allowed is easily characterized as the number of different rows in the matrix M_f associated with f. In a seqeunce of results by Papadimitriou and Sipser [15], Duris et al. [16], Nisan and Widgerson [17] it was proved that for every $k \geq 1$ there are functions such that computing them with a k-round protocol is exponentially more expensive than computing the same functions with a $k+1$-round protocol.

Randomized protocols. In the basic model of communication complexity described in the introductive section, Alice and Bob are deterministic. This means that in any stage, when one of the two players needs to send a bit, the value of the bit is a deterministic function of the player's input and the communication so far. In the randomized model, we allow Alice and Bob to "toss coins" during the execution of the protocol and take into account the outcome of the coin tosses when deciding what messages to send. This implies that the communication on a given input (x, y) is not fixed any more, but becomes a random variable. Similarly the output computed by a randomized protocol on input (x, y) is also a random variable. Roughly randomized protocols can fall into two categories. The first category incudes only protocols that always output the correct value. The second category includes protocols that may err, but for every input (x, y) they are guaranteed to compute the correct value $f(x, y)$ with high probability. Also the cost of a randomized protocol can be defined in two ways. We can analyze the worst case behaviour of a protocol or we can analyze the average case behaviour. Randomized communication complexity was defined by Yao [1] in his seminal paper.

Amortized Communication Complexity. One can ask whether computing a function f on l instances can be done more efficiently than just computing f on each of the l instances separately. For instance, suppose you want to compare l files (each file is a string of length n) $x_1, x_2, \ldots, x_l$ which are stored in a site with other l files $y_1, y_2, \ldots, y_l$ which are stored in another site. It is interesting to decide whether this can be done using less than ln bits, that is the number of bits required by using the protocol of the function Eq on each of the l files (x_i, y_i). It seems very interesting to answer the following question: "Can we solve l problems simultaneously in a way that is better than solving each of the l problems separately was raised in the work of [18].

Relations. In this paper we concentrated in computing functions. We can generalize it to the task of computing relations. In this generalized model, given an input (x, y), Alice and Bob need to find an output z such that x, y and z satisfy some relation R. For instance they may want to know an index i such that the i-th components of x and y are different. This generalization of communication complexity to the case of relations was first discussed in [19]. Their motivation was the connection between the communication complexity of a certain type of relations and the complexity of Boolean circuits.

Multiparty protocols. In the models considered so far there were only two parties. We can extend the model to the case where there are $k > 2$ parties that want to compute $f(x_1, x_2, \ldots, x_k)$. The natural extension is to consider k players $P_1, P_2, \ldots, P_k$, such that P_i only knows x_i. The exact form of communication between the k players must be specified. One possibility is to assume that every message sent by one of the players is seen by all the others (broadcast transmission). Another possiblily is to assume that the message sent by one of the players is seen only by a limited number of other players (in a very restricted model we can assume that the message of P_i can be seen only by P_{i+1}). Several models for multiparty communication were studied in the literature. See for instance [20]. Also see [21], [22] for applications of this model.

Variable partition models. In the standard two-party model the input (x, y) is partitioned in a fixed way, i.e Alice always gets x and Bob always gets y. In the variable partition model the input (x, y) is seen as a string of $2n$ bits (n bits of x and n bits of y) and the two parties are allowed to choose the partition of the $2n$ bits. This partition is chosen based on f, regardless of the specific input. We have seen in the previous section that $Eq(x, y)$ requires $n + 1$ bits of communication in the worst case. On the other side if Alice gets $x_1, \ldots, x_{\frac{n}{2}}, y_1, \ldots, y_{\frac{n}{2}}$ and Bob gets $x_{\frac{n}{2}+1}, \ldots, x_n, y_{\frac{n}{2}+1}, \ldots, y_n$, then they can compute $Eq(x, y)$ by exchanging only two bits with the following protocol. Alice checks that the first half of the string x is equal to the first half of the string y. If they are different she sends the bit 1 to Bob and they stop to interact. If they are equal she sends the bit 0 to Bob that checks if the second half of x is equal to the second half of y. After checking, Bob sends to Alice the bit 0 if they are equal and the bit 1 if they are different. The best case partition model introduced in [14] defines $D^{best}(f)$ as the deterministic communication complexity of the best protocol for computing the function f with respect to the best partition of the bits of the input pair (x, y). Obviously, $D^{best}(f) \leq D(f)$. The best case partition model is useful in many cases that we may choose the locations in that input variables are accessed.

6 Applications

In this last part of the paper we show some applications of the results about communication complexity. A first and immediate application is to the management of a distributed system. In such a system it is often required to check whether two copies of a file that reside in two differents sites are the same. Clearly, this is just computing the function Eq whose communication complexity has been studied in the previous section. The two parties of the communication are the programs residing in the two different sites. Their inputs can be imagined as an agreed binary codification of the respective files.

Another application of the communication complexity is to prove time lower bounds for networks. Suppose to have a network consisting of k processors. Each

processor P_i of the network gets as an input some value x_i and together they wish to compute the value of $f(x_1, x_2, \ldots, x_k)$. There are essentially two lower bounds tecniques for such networks. The first is the network diameter lower bound. It states that information must travel from one end of the network to the other hand, thus the computation time requires Ω (diameter of the network) time. The other method makes use of communication lower bounds. Kutshilevitz and Nisan described this method in [2]. The method consists of two stages. First we partition the network into two large parts such that the number of the edges connecting the two parts (sometimes called bandwidth) is small. A known result of [5] states that any planar network of p processors can be partitioned in two parts, each containing at least $p/3$ processors, such that the number of edges connecting the two parts is $O(p^{1/2})$. In the second stage we view the computation done by the network as a two party communication problem, where the input of each party is all the inputs given to the processors in one of the parts of the network. If the two party communication problem requires to submit at least d bits, then this implies that the time it takes for the network to complete this computation is at least d divided by the bandwidth (the time it takes to submit a single bit over an edge is the time unit). Leighton [6] applied this method to prove lower bounds in various types of networks.
We have just considered the time it takes to compute the function f on the network, ignoring the number of bits transmitted in each time step. It maybe also interesting to ask what is the total number of bits exchanged between the processors of the network. This question was first posed by Tiwari in [9]. Of a particular interest among researchers is the *line network*, consisting of $k+1$ processors $P_0, P_1, \ldots, P_k$ with edges only between P_i and P_{i+1}, for $0 \leq i \leq k-1$. The processors wish to compute a function $f(x, y)$ where x is stored in P_0 and y is stored in P_k. The complexity of a protocol is the total number of bits exchanged on all edges. Denote such number by $D_k(f)$. It is easy to see that $D_k(f) \leq kD(f)$. This is true if we think that the processor P_0 can simulate Alice, the processor P_k can simulate Bob, and the intermediate processors behave as a relay, i.e they simply propagate the messages they receive. The main question is whether we can do better: is it true that for every function f, $D_k(f) = \Omega(kD(f))$?

Another interesting and well studied application of communication complexity is to VLSI chips. This was also the first motivation for studying communication complexity [24]. A VLSI chip can be viewed as an $a \times b$ grid which has n input ports and one output port. There are gates on some vertices of the grid and wires connecting gates with other gates or with the ports. The two most important complexity measures for a VLSI chip are its area $A = a \times b$ and the time T it takes from the time the input is provided in the input ports to the time the results appear in the output port. The designer of a VLSI chip that computes a function $f(x_1, x_2, \ldots, x_m)$ must decide which gates to use, how to connect them and which input x_i should be fed into each input port. A known result gives a way to prove lower bounds on the quantity AT^2 using the measure $D^{best}(f)$ discussed in the previous section. It states that if we have a VLSI chip with area A and time T, then $D^{best}(f) \leq \sqrt{AT}$.

References

[1] A.Yao. Some complexity questions related to distributive computing. Proceedings, 11th Annual ACM Symposium on Theory of Computing, 1979, pag. 209-213.

[2] E.Kushilevitz and N.Nisan. Communication complexity. Cambridge University Press, 1997.

[3] K.Mehlhorn and E.Schmidt. Las Vegas is better than Determinism in VLSI and Distributed Computing. Proceedings of 14th ACM Symposium on Theory of Computing, 1982, 330-337.

[4] R.J.Lipton and R.Sedgewick. Lower bounds for VLSI. Proceedings of 13th ACM Symposium on Theory of Computing, 1981, 300-307.

[5] R.J.Lipton and R.E.Tarjan. Applications of a Planar Separator Theorem. Siam J. Computing, Vol.9, pag.615-627, 1980.

[6] F.T.Leighton. Introduction to Parallel Algorithms and Architectures: Arrays, Trees,, Hypercubes. Morgan-Kaufmann, 1991

[7] L.Lovasz and M.Saks. Lattices, Möbius Functions and Communication Complexity. Journal of Computer and System Sciences 47, 1993, 322-349.

[8] J.Sgall. Private communication to E.Kushilevitz and N.Nisan transmitted through the book "Communication Complexity".

[9] P.Tiwari. Lower bounds on Communication complexity in Distributed Computer Networks. Journal of ACM 34 (4), 1987, 921-938.

[10] N.Alon and P.D.Seymour. "A Counterexample to the Rank-Coloring Conjecture". Journal of Graph Theory 13, 1989, 523-525.

[11] A.A.Razborov. "On the Distributional Complexity of Disjointness". Theoretical Computer Science 106 (2), 1992, 385-390.

[12] R.Raz and B.Spieker. "On the *logrank* Conjecture in Communication Complexity". Combinatorica 15(4), 1995, 567-588.

[13] N.Nisam and A.Widgerson. "On rank vs. Communication Complexity". Combinatorica, 15(4), 1995, 557-566.

[14] A.Aho, J.Ullman and M.Jannakakis. "On notions of Information transfer in VLSI Circuits". Proceedings of 15th STOC, 1983, 133-139.

[15] C.Papadimitriou and M.Sipser. "Communication complexity". JCSS, 28, No.2, 1984, 260-269.

[16] P.Duris, Z.Galil and G.Shnitger. "Lower bounds on Communication Complexity". Information and Computation, 73, No.1,1987, 1-22

[17] N.Nisan and A.Wigderson. "Round in Communication Complexity revisited". Siam J.Computing, 22, No.1, 1993, 211-219.

[18] M.Karchmer, R.Raz and A.Wigderson. "On proving super-Logarithmic Depth Lower Bounds via the Direct Sum in Communication Complexity". Proceedings of the 6th IEEE Structure in Communication Complexity Theory ,1991, 299-304.

[19] M.Karchmer and A.Wigderson. "Monotone circuits for Connectivity Require SuperLogarithmic Depth". Siam J.Discrete Mathematics, 3(2), 1990, 255-265.

[20] D.Dolev and T.Feder. "Multiparty Communication Complexity". Proceedings of 30th IEEE Symposium on Foundations of Computer Science, 1989, 428-433.

[21] A.K.Chandra, M.L.Furst and R.J.Lipton. "Multiparty protocols". Proceedings of 15th STOC, 1983, 94-99.

[22] L.Babai, N.Nisan and M.Szegedy. "Multiparty protocols, Pseudorandom Generators for LOGSPACE, and Time-Space Trade-offs". JCSS, 45, No.2, 1992, 204-232.

[23] L.Babai and P.Frankl and J.Simon. "Complexity classes in communication complexity theory". Proceedings of the 27th IEEE Symposium on Foundations of Computer Science, 1986, 337-347.

[24] C.D.Thompson. "Area-Time complexity for VLSI". Proceedings of 11th ACM Symposium on Theory of Computing, 1979, 81-88.