INFORMATIQUE THÉORIQUE ET APPLICATIONS

JÁN MAŇUCH

Construction of very hard functions for multiparty communication complexity

Informatique théorique et applications, tome 34, nº 1 (2000), p. 61-75

<http://www.numdam.org/item?id=ITA_2000__34_1_61_0>

Theoretical Informatics and Applications
Theoret. Informatics Appl. **34** (2000) 61–75

CONSTRUCTION OF VERY HARD FUNCTIONS FOR MULTIPARTY COMMUNICATION COMPLEXITY*

JÁN MAŇUCH[1]

Abstract. We consider the multiparty communication model defined in [4] using the formalism from [8]. First, we correct an inaccuracy in the proof of the fundamental result of [6] providing a lower bound on the nondeterministic communication complexity of a function. Then we construct several very hard functions, *i.e.*, functions such that those as well as their complements have the worst possible nondeterministic communication complexity. The problem to find a particular very hard function was proposed in [7], where it has been shown that almost all functions are very hard. We also prove that combining two very hard functions by the Boolean operation xor gives a very hard function.

AMS Subject Classification. 68Q15, 68Q85.

INTRODUCTION

The multiparty model is a natural extension of the two-party model. The aim is to compute a given Boolean function on an input. In the two-party communication model, the input is distributed between two parties, which are connected by a communication link. The communication complexity is the total amount of communication on the link needed to compute the function. Naturally, the goal is to compute the function minimizing the communication complexity.

In the multiparty model the input is distributed among n parties. It is assumed that there is a coordinator that is allowed to communicate to each party, but the parties are not allowed to communicate directly with each other. The communication complexity is the total amount of communication on all links. The goal is the same: to compute a function on the whole input minimizing the communication complexity.

---

* *Supported by Academy of Finland under Grant No. 44087.*

[1] Department of Mathematics and Turku Centre for Computer Science, University of Turku, Turku, Finland; e-mail: manuch@cs.utu.fi

One of the trivial solutions how to compute a Boolean function is sending all data in the parties to the coordinator which will perform all computations. Hence, the worst communication complexity of a function is equal to the size of the input. The Boolean function which can be computed only with the worst nondeterministic communication complexity is called *hard function.* If the complement of such a function is also a hard function, then we call this function *very hard.* The aim of this paper is to construct several concrete very hard functions.

The two-party model has been extensively studied; an overview of results on the two-party model can be found in [8]. The study of the communication complexity of the two-party model was inspired by the VLSI circuits complexity, *cf.* [10] and [13]. There are many other applications of the communication complexity, *cf.* [9]. The multiparty model was introduced and investigated in [4].

Note that in [3] a different multiparty model was considered. In that model each of the n parties has all the inputs except one, and all parties communicate through a shared "blackboard". This model was investigated in [1], where an interesting relation to time-space tradeoffs and branching programs was discovered. We do not know any connections between these two models.

There are only few results known about the multiparty model introduced in [4]. In [5] an upper bound on the deterministic communication complexity of order $O(\mathrm{k}(f)\,\mathrm{k}^2(1-f)\,\mathrm{ncc}(f)\,\mathrm{ncc}(1-f))$ was established, where $\mathrm{k}(f)$ is the number of processors accessed and $\mathrm{ncc}(f)$ is the nondeterministic communication complexity of the function f. In [6] a fundamental tool to prove a lower bound on the communication complexity of a given Boolean function was developed. However, there was a small mistake in the proof. In this paper we correct the proof of this result. Further results proved in [6] are the following relations between the nondeterministic communication complexities of a function and its complement, and also between the deterministic and nondeterministic communication complexities:

$$\mathrm{ncc}(1-f) \leq n\left(1+2^{\mathrm{ncc}(f)}\right), \quad \mathrm{cc}(f) \leq n\left(1+2^{\mathrm{ncc}(f)}\right) \cdot$$

For the restricted one-way model, where only one communication is allowed for each direction on each link, the following results for one-way nondeterministic and one-way deterministic communication complexity were established in [6]:

$$\mathrm{ncc}_1(f) = \mathrm{ncc}(f), \quad \mathrm{cc}_1(f) \leq \mathrm{cc}(f) \cdot 2^{\mathrm{cc}(f)+1}.$$

For all these bounds it is shown that they are optimal.

The main result of this paper is the construction of several very hard functions. Recall that a function f is a very hard function if f and its complement $1-f$ have the worst nondeterministic communication complexity. Of course, in the deterministic case the communication complexities of a function f and its complement $1-f$ are equal. In the nondeterministic case there could be even an exponential difference between the communication complexities of f and $1-f$, *cf.* [6]. In [7] it has been shown that almost all functions are very hard, while finding a particular very hard function was proposed as an open problem.

Note that a similar problem appears in the theory of circuit size complexity. It is well-known fact that almost all Boolean functions of n variables require $\Omega(2^n/n)$ combinatorial circuit complexity, *cf.* [12] and [11]. Here, the combinatorial circuit complexity of a Boolean function f is the minimal number of gates needed to realize the function f, where gates are any of all 16 binary operations. On other hand, the highest known lower bound of combinatorial circuit complexity of a concrete function is only $3n$, *cf.* [2].

We also prove that combining two very hard functions by the Boolean operation *xor* results in a new very hard function. This result extends a similar result from [6] that claims that the Boolean operation *and* preserves the hard functions and the result from [7] that claims that the deterministic communication complexity adds up when combining two functions by xor.

1. Preliminaries

In this section we define the multiparty model, the nondeterministic protocols and the communication complexity. We start with an informal definition of the model.

The multiparty model consists of a coordinator and n parties. The coordinator wishes to evaluate a Boolean function $f(x_1, \ldots, x_n)$. The input vector $x = (x_1, \ldots, x_n)$ is distributed among n parties, with $x_i \in \{0,1\}^m$ known to the party i. We allow a communication only between the coordinator and any party. Instead of saying "the communication between the coordinator and party i" we often say "the communication on link i". The computation consists of several phases, where one phase is as follows:
The coordinator sends some nonempty messages to some parties and then, each party that got a message, sends a nonempty message back to the coordinator. After the computation the coordinator announces the result: $\bar{1}$ if accepted or $\bar{0}$ if rejected. In the nondeterministic case we accept an input, if there exists an accepting computation for this input.

Next, we give formal definitions of the nondeterministic protocol. We will use the notation from [8]. Let λ be the empty string. In the following we identify a relation $\Phi \subseteq A \times B$ with the function $\Phi : A \to 2^B$ defined as follows: for every $a \in A$, $\Phi(a) = \{b \mid (a,b) \in \Phi\}$.

Definition 1.1. A *nondeterministic protocol* over $X = \{x_1, \ldots, x_n\}$ with $x_i = x_i^1 \ldots x_i^m \in \{0,1\}^m$, $n \geq 1$, $m \geq 2$, is a pair $P = \langle \Phi_C, \Phi_P \rangle$, where

(a) Φ_C is a communication relation of the coordinator in

$$[\{0,1,\$\}^*]^n \times ([\{0,1\}^*]^n \cup \{\bar{0}, \bar{1}\});$$

and the projections of Φ_C are defined as follows: for all $c \in [\{0,1,\$\}^*]^n$, $i \in \{1, \ldots, n\}$ let

$$\Phi_{C,i}(c) = \{d_i \mid (d_1, \ldots, d_n) \in \Phi_C(c) \cap [\{0,1\}^*]^n\};$$

(b) Φ_P is a communication relation of parties in

$$(\{1,\dots,n\} \times \{0,1\}^m \times \{0,1,\$\}^+) \times \{0,1\}^+,$$

where

(i) Φ_C has the *prefix freeness property*:
for each $i \in \{1,\dots,n\}$, $c,c' \in [\{0,1,\$\}^*]^n$, $d,d' \in \{0,1\}^+$: let c_i (let c'_i) be the i-th component of c (of c');
if $c_i = c'_i$ and $(c,d),(c',d') \in \Phi_{C,i}$, then d is not any proper prefix of d';
if c_i is a proper prefix of c'_i, *i.e.*, $c'_i = c_i d' \$ m$ with $m \in \{0,1,\$\}^*$ and $(c,d) \in \Phi_{C,i}$, then d is not any proper prefix of d' and *vice versa* (each party knows when it is the end of a message from the coordinator),

(ii) Φ_P has the *prefix freeness property*:
for each $i \in \{1,\dots,n\}$, $c \in \{0,1,\$\}^*$, $d,d' \in \{0,1\}^+$ and $x_i, x'_i \in \{0,1\}^m$ such that $((i,x_i,c),d),((i,x'_i,c),d') \in \Phi_P$, d is not any proper prefix of d' (the coordinator knows when it is the end of a message from any party),

(iii) for each $i \in \{1,\dots,n\}$, $c,c' \in [\{0,1,\$\}^*]^n$ such that $c_i = c'_i \neq \lambda$, if $\Phi_C(c) \cap \{\bar{0},\bar{1}\} \neq \emptyset$, then $\Phi_C(c') \subseteq \{\bar{0},\bar{1}\}$ or $\Phi_{C,i}(c') = \{\lambda\}$ (if there is a communication on link i then party i knows when it is finished).

Let us give an intuitive explanation of the symbols used in the above definition. $0,1$ represent bits sent through the communication links and \$ is a virtual end mark of messages. *Virtual* means that the symbol is not send neither by the coordinator, nor by any party. The properties *(i)–(iii)* ensure that any such virtual end mark is not necessary. In fact, the use of such a mark during the communication can lead to very nonintuitive properties of the model: consider, for example, a node which sends k empty messages, and in this way the receiver obtains an arbitrary large information, the number k, without increasing the communication complexity. Symbols $\bar{0},\bar{1}$ represent the result value of a computation.

The relation Φ_C maps a temporary state of a communication on all links to the one of the result values or to an n-ary sequences of messages sent from the coordinator to the each party. The empty message means that the coordinator does not communicate with the party. The relation Φ_P maps the number i of a party, the input of the party i and a temporary state of the communication on the link i to a nonempty message sent from the party i back to the coordinator. Since we consider nondeterministic protocols, these maps could be ambiguous.

We proceed with the definition of the computation.

Definition 1.2. A *computation* of a protocol $P = \langle \Phi_C, \Phi_P \rangle$ on an input vector $x = (x_1,\dots,x_n)$ is a communication vector $c = (c_1,\dots,c_n)$ with

$$c_i = c_i^1 \$ c_i^2 \$ \dots \$ c_i^{2r_i-1} \$ c_i^{2r_i} \$,$$

where

(i) for all $i \in \{1,\dots,n\}$ $c_i^1,\dots,c_i^{2r_i} \in \{0,1\}^+$;

(ii) there is an integer r, called the *number of rounds* of c, and a sequence of vectors $c_{[0]}, \ldots, c_{[2r]} \in [\{0,1,\$\}^*]^n$, *i.e.*, states of the computation such that

(a) $c_{[0]} = (\lambda, \ldots, \lambda)$,

(b) if l is even, then

$$c_{[l+1]} \in \{(c_{[l]1} d_1 \$, \ldots, c_{[l]n} d_n \$) \mid (d_1, \ldots, d_n) \in \Phi_C(c_{[l]})\},$$

(c) if l is odd, then for each $i \in \{1, \ldots, n\}$

$$\begin{aligned} c_{[l+1]i} &= \lambda, && \text{if } c_{[l]i} = \$, \\ c_{[l+1]i} &= c'\$, && \text{if } c_{[l]i} = c'\$\$, \\ c_{[l+1]i} &\in \{c_{[l]i} d\$ \mid d \in \Phi_P(i, x_i, c_{[l]i})\}, && \text{otherwise,} \end{aligned}$$

(d) $c_{[2r]} = c$;

(iii) $\Phi_C(c) \subseteq \{\bar{0}, \bar{1}\}$.

Informally, the vectors $c_{[2j-1]}$ and $c_{[2j]}$ represent the states of computation in the round j. Vector $c_{[2j-1]}$ is the state after the coordinator sends messages to chosen parties. If the coordinator does not communicate with a party i in the round j, then the i-th component of $c_{[2j-1]}$ contains two symbols \$ in the end. Hence the second one is removed by the definition in part *(ii.c)*, when computing $c_{[2j]}$, *i.e.*, the state after the chosen parties send back messages to the coordinator.

We denote the set of all computations on an input x (all computations) by $\text{comp}(P, x)$ (by $\text{comp}(P)$). We say that a computation c is *accepting*, if $\bar{1} \in \Phi_C(c)$. P is called an *r-round nondeterministic protocol* if every computation of P has at most r rounds.

We say that P *computes* 1 for an input vector x, *i.e.*, $P(x) = 1$, if there exists an accepting computation c of P on x, otherwise P computes 0, *i.e.*, $P(x) = 0$. We say that P *computes* the Boolean function f with the input variables X, if for each $x \in [\{0,1\}^m]^n$ we have $f(x) = P(x)$.

Now, we illustrate the above definitions with the following example.

Example 1.3. Consider the function f defined by

$$f(x_1, \ldots, x_n) = 1 \text{ iff there exist } i \neq j \text{ such that } x_i = x_j.$$

We construct a nondeterministic protocol $P = \langle \Phi_C, \Phi_P \rangle$ computing this function:

$$\Phi_C(\lambda, \ldots, \lambda) = \{(b_1, \ldots, b_n) \mid b_i = b_j \in \{0,1\} \text{ and } \forall k \neq i, j : \ b_k = \lambda\},$$

$$\Phi_P(i, x_i, b\$) = \begin{cases} \{x_i^2 \ldots x_i^m\}, & \text{if } x_i^1 = b, \\ \emptyset, & \text{otherwise,} \end{cases}$$

$$\Phi_C(c_1, \ldots, c_n) = \begin{cases} \bar{1}, & \text{if } c_i = c_j = b\$y\$,\ b \in \{0,1\},\ y \in \{0,1\}^{m-1} \\ & \text{and } \forall k \neq i, j\colon\ c_k = \lambda, \\ \emptyset, & \text{otherwise.} \end{cases}$$

Φ_C has the prefix freeness property, since all messages the coordinator sends, have the same length. The same holds for Φ_P. If we have $c, c' \in [\{0,1,\$\}^*]^n$ with $c_i = c'_i \neq \lambda$ and $\Phi_C(c) \cap \{\bar{0}, \bar{1}\} \neq \emptyset$, then $\Phi(c') \subseteq \{\bar{0}, \bar{1}\}$, so P satisfies also condition *(iii)* of Definition 1.1. Hence it is a nondeterministic protocol.

Informally we can describe the protocol P as follows: the coordinator guesses for which $i \neq j$ the equality $x_i = x_j$ holds. It also guesses the first bit of $x_i = x_j$ and sends it to the parties i and j. It does not communicate with the other parties at all. The party i (the party j) checks if the coordinator guessed the first bit of x_i (of x_j) correctly and in such case sends back the rest of the input x_i (of the input x_j). Finally, the coordinator checks if the rest of x_i equals to the rest of x_j. Only in such case it ends with $\bar{1}$. Clearly, this protocol computes the function f.

Consider an input vector $x = (x_1, \dots, x_n)$ such that $x_1 = x_2 = a_1 \dots a_m \neq x_3$ and $a_1 = x_3^1$. Then

$$c = (a_1\$a_2 \dots a_m\$, a_1\$a_2 \dots a_m\$, \lambda, \dots, \lambda)$$

is an accepting computation of P on x. Indeed, we have $r_1 = r_2 = 1$, $r_3 = \dots = r_n = 0$ and the sequence of states of the computation is:

$$c_{[0]} = (\lambda, \dots, \lambda), \qquad c_{[1]} = (a_1\$, a_1\$, \$, \dots, \$),$$
$$c_{[2]} = (a_1\$a_2 \dots a_m\$, a_1\$a_2 \dots a_m\$, \lambda, \dots, \lambda) = c.$$

The number of rounds of c is 1. The computation

$$c' = (a_1\$a_2 \dots a_m\$, \lambda, a_1\$x_3^2 \dots x_3^m\$, \lambda, \dots, \lambda)$$

is not any accepting computation. Obviously, each computation uses exactly 1 round, so P is a 1-round protocol.

Now we look at some simple properties of computations. Conditions *(i)–(iii)* ensure that the set of all computations $\mathrm{comp}(P)$ has the *self-delimiting* property:

Lemma 1.4. *Let $i \in \{1, \dots, n\}$, $c, c' \in \mathrm{comp}(P)$ be computations such that $c_i \neq c'_i$ are both nonempty and q be the largest number such that $c_i^1 = {c'_i}^1, \dots, c_i^q = {c'_i}^q$. Let r_i (let r'_i) be the number of rounds, in which there was a communication on the link i in computation c (in computation c'). Then*

$$q < 2r_i, \; q < 2r'_i, \tag{SD1}$$
$$c_i^{q+1} \textit{ is not any proper prefix of } {c'_i}^{q+1}. \tag{SD2}$$

Let us define the morphism h, which deletes the virtual mark \$: $h(0) = 0$, $h(1) = 1$, $h(\$) = \lambda$. So we have $h(c_i) = c_i^1 \dots c_i^{2r_i}$. Moreover, we define $h(c) = h(c_1) \dots h(c_n)$. We can state a corollary of Lemma 1.4.

Corollary 1.5. [6] *Let $c, c' \in \mathrm{comp}(P)$ and assume c_i, c'_i are both nonempty, then $h(c_i)$ is not any proper prefix of $h(c'_i)$.*

Sometimes we are not interested in the communication on all links, but only in the communication on selected links. We use the following notation:

Notation 1.6. Let $J = \{j_1, \ldots, j_k\} \subseteq \{1, \ldots, n\}$ be a nonempty set and let $c = (c_1, \ldots, c_n)$ be a computation. Then $c/J = (c_{j_1}, \ldots, c_{j_k})$.

We finish this section by the definition of the nondeterministic communication complexity.

Definition 1.7. The *length of a computation* c of P is the total length of all messages $m \in \{0,1\}^+$ in c. For each $x \in [\{0,1\}^m]^n$ such that $P(x) = 1$, let $\mathrm{ncc}(P, x)$ denote the length of the shortest accepting computation of P on x. The *nondeterministic communication complexities* of the protocol P and of the function f are

$$\mathrm{ncc}(P) = \max\{\mathrm{ncc}(P, x) \mid P(x) = 1\},$$
$$\mathrm{ncc}(f) = \min\{\mathrm{ncc}(P) \mid P \text{ computes } f\} \cdot$$

Similarly, we can define $\mathrm{ncc}(P, x/J)$, $\mathrm{ncc}(P/J)$ and $\mathrm{ncc}(f/J)$, when we measure only the communication on links in J.

Example 1.8 (continued). The length of both computations c and c' is $2m$. Clearly this holds for any accepting computation of P. Hence $\mathrm{ncc}(P) = 2m$, which implies $\mathrm{ncc}(f) \leq 2m$.

Using the idea of the protocol P for any Boolean function f we can construct a protocol with the nondeterministic communication complexity nm. First, the protocol sends all the information the parties have to the coordinator and then the coordinator is able to compute f. This implies:

Proposition 1.9. *For any Boolean function* $f : [\{0,1\}^m]^n \to \{0,1\}$, $\mathrm{ncc}(f) \leq nm$.

Now we can define *hard* and *very hard* functions.

Definition 1.10. A Boolean function is *hard*, if $\mathrm{ncc}(f) = nm$ and it is *very hard*, if $\mathrm{ncc}(f) = \mathrm{ncc}(1 - f) = nm$.

2. The fundamental tool

In this section we correct Theorem 1 and its proof from [6]. The mistake in [6] occurs actually in Lemma 1. Here is the correct version of Lemma 1.

Lemma 2.1. *Let* $b_1, b_2, \ldots, b_p$ *be a sequence of nonempty binary strings such that no element is a proper prefix of another and no element of this sequence occurs in it more than* q *times. Then* $\sum_{i=1}^{p} |b_i| \geq p \log \frac{p}{q}$.

Before the proof let us show that Lemma 1 in [6] does not hold.

Example 2.2. In Lemma 1 and its proof of [6] the inequalities $\sum_{i=1}^{p} |b_i| \geq p \log \left\lceil \frac{p}{q} \right\rceil$ and $\sum_{i=1}^{p} |b_i| \geq p \left\lceil \log \frac{p}{q} \right\rceil$ are claimed, and the later one is used in the proof of Theorem 1 in [6]. However, the following sequence of binary strings 00, 00, 00, 01, 1, 1, 1 shows that neither of these is true. Indeed, we have $\sum_{i=1}^{p} |b_i| = 11$, $p = 7$ and if we choose $q = 3$, then $p \log \left\lceil \frac{p}{q} \right\rceil \doteq 11,09$ and $p \left\lceil \log \frac{p}{q} \right\rceil = 14$.

Proof of Lemma 2.1. Let $q' \leq q$ be the maximal number of occurrences of a string in the sequence. For $i = 1, \ldots, q'$ consider sets Q_i containing strings which occur in the sequence at least i times. Clearly, $\sum_{i=1}^{q'} |Q_i| = p$. For each set Q_i there is a corresponding binary tree such that the strings in Q_i encodes the paths of the tree. Since no element in the sequence is a proper prefix of another element, all strings end in the leaves of the tree. The trees $Q_1, \ldots, Q_{q'}$ form together the forest G. Let $T(G)$ denote the total sum of the depths of all leaves of the forest G. Obviously, $T(G) = \sum_{i=1}^{p} |b_i|$. Now we transform the forest G into a forest G' in two steps:

Step (i). Apply the following procedure as many times as possible. If some node in of the forest has only one son, then we delete the node and replace it by its son. In this way we get a forest containing only binary trees.

Step (ii). Apply the following procedure as many times as possible. Let v_1 (v_2) be a leaf with the maximal (minimal) depth. If $\text{depth}(v_1) \leq \text{depth}(v_2) + 1$, then the forest is balanced. Otherwise, we perform the following operation: if v is the father of v_1, then we cut off both sons of v and connect them to v_2. In this way we get the balanced forest G' with depths of leaves either h or $h + 1$.

Note that in both steps we preserve the numbers of both leaves and trees, and the depths of leaves can only decrease, thus $T(G') \leq T(G)$. Let a (let b) be the number of leaves in depth h (in depth $h + 1$). Clearly, $a + b = p$. It is easy to derive $q' \cdot 2^{h+1} = 2a + b = p + a$, which implies

$$T(G) \geq T(G') = ha + (h+1)b = p \log \frac{p+a}{q'} - a \geq p \log \frac{p}{q'},$$

where the last inequality comes from the inequality $\log(x + 1) \geq x$ for $x \in \langle 0, 1 \rangle$, in which we substitute $x = \frac{a}{p}$. □

To state the theorem we need another two definitions.

Definition 2.3. [6] Let Y be a nonempty subset of $f^{-1}(1)$. We say that the index j is *important for f with respect to Y*, if for every $y = (y_1, \ldots, y_n) \in Y$ there is $y'_j \in \{0,1\}^m$ such that

$$f(y_1, \ldots, y_{j-1}, y'_j, y_{j+1}, \ldots, y_n) = 0.$$

Let $J \subseteq \{1, \ldots, n\}$ be a nonempty set of indices and $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_n)$ two input vectors. We denote

$$[x \cdot y]/J = (z_1, \ldots, z_n), \text{ where } z_i = \begin{cases} x_i, & \text{if } i \in J, \\ y_i, & \text{otherwise.} \end{cases}$$

Let M be a nonempty sets of inputs. Then *J-closure of M* is the set

$$C\ell_J(M) = \{[x : y]/J \mid x, y \in M\} \cdot$$

If we use Lemma 2.1 instead of incorrect Lemma 1 in [6] in the proof of the theorem, we get the following result.

Theorem 2.4. *Let Y be a nonempty subset of $f^{-1}(1)$ and $J_1, \ldots, J_r$ be pairwise disjoint sets of indices. Assume that every $j \in \cup_{i=1}^r J_i$ is an important index for f with respect to Y and let d_i be positive integers satisfying*

$$d_i \geq \max_{M \subseteq Y} \{|M| \mid C\ell_{J_i}(M) \subseteq f^{-1}(1)\}$$

Then we have lower bounds on the nondeterministic communication complexity

$$\mathrm{ncc}(f/J_i) \geq \log \frac{|Y|}{d_i}, \qquad \mathrm{ncc}(f) \geq \sum_{i=1}^{r} \log \frac{|Y|}{d_i} \cdot$$

The proof is based on Lemma 2.1, properties of nondeterministic protocols (Cor. 1.5) and the classical crossing sequence argument, *cf.* [6]. Note that the above theorem differs from Theorem 1 in [6], where it was claimed $\mathrm{ncc}(f) \geq \sum_{i=1}^r \left\lceil \log \frac{|Y|}{d_i} \right\rceil$. We apply this theorem in Example 1.3.

Notation 2.5. For every $a \in \{0,1\}^m$ and every integer d let $\mathrm{val}(a)$ denote the value of the binary string a and let $(a)_d$ denote the string from $\{0,1\}^m$ such that $\mathrm{val}((a)_d) \equiv \mathrm{val}(a) + d \bmod 2^m$.

Example 2.6 (continued). Assume that $n \leq 2^{m-1} + 1$. Let us choose in Theorem 2.4: $r = 2$, $J_1 = \{1\}$, $J_2 = \{2\}$ and $Y = \{(a, a, (a)_1, (a)_2, \ldots, (a)_{n-2}) \mid a \in \{0,1\}^m\}$. It is easy to see that indices 1 and 2 are important with respect to Y. Take arbitrary different $a, b \in \{0,1\}^m$. Assume that $f(a, b, (b)_1, (b)_2, \ldots) = f(b, a, (a)_1, (a)_2, \ldots) = 1$. Then we have $a = (b)_c$ and $b = (a)_d$ for some integers $1 \leq c, d \leq n - 2 \leq 2^{m-1} - 1$. Obviously $c + d = 2^m$, which contradicts the above restrictions for c and d. Hence we can set $d_1 = 1$ and similarly $d_2 = 1$. Using Theorem 2.4 we obtain $\mathrm{ncc}(f) \geq 2\log(2^m) = 2m$. Thus we can conclude $\mathrm{ncc}(f) = 2m$ for all $n \leq 2^{m-1} + 1$.

3. Very hard functions

In this section we give the main result of the paper. We present two methods how to construct a very hard function for any n using a strongly very hard function

$f(x_1, x_2)$ or using very hard function(s) with less than n variables. We also show that the function $g(x_1, x_2) = 1$ iff $\mathrm{val}(x_1) \leq \mathrm{val}(x_2)$ is very hard and a simple modification of the function g, denoted below by $g'(x_1, x_2)$, is strongly very hard. For this purpose we need some more definitions.

Definition 3.1. Let $f(x_1, \ldots, x_n)$ be a Boolean function, $x_i \in \{0,1\}^m$ and let $J_i = \{i\}$ for all $i \in \{1, \ldots, n\}$. If there exists a nonempty set $Y \subseteq f^{-1}(1)$ such that for all $i \in \{1, \ldots, n\}$ the index i is important with respect to Y and there exist positive integers d_i such that for every $i \in \{1, \ldots, n\}$

$$d_i \geq \max_{M \subseteq Y}\{|M| \mid C\ell_{J_i}(M) \subseteq f^{-1}(1)\}, \text{ and} \tag{1}$$

$$\log \frac{|Y|}{d_i} = m, \tag{2}$$

then the function f is *strongly hard.* We will say that the function f is *strongly very hard* when both the function f and its complement $1 - f$ are strongly hard.

According to Theorem 2.4 it is obvious that any strongly (very) hard function is also a (very) hard function. First, we modify the function $g(x_1, x_2)$ and prove that the resulting function $g'(x_1, x_2)$ is strongly very hard. Let

$$g'(x_1, x_2) = 1 \text{ iff } \mathrm{val}(x_1) \leq \mathrm{val}(x_2) \wedge (x_1, x_2) \neq (0^m, 1^m) \vee (x_1, x_2) = (1^m, 0^m).$$

Theorem 3.2. *The function g' is strongly very hard.*

Proof. First, we check the function g'. Set $J_1 = \{1\}$, $J_2 = \{2\}$, $Y = \{(x_1, x_2) \mid x_1 = x_2\}$. Take any vector $x = (a, a) \in Y$. We have $g'((a)_{+1}, a) = g'(a, (a)_{-1}) = 0$, hence both indices are important with respect to Y. Consider now $a \neq b$. Clearly, exactly one of the values $g'(a, b)$ and $g'(b, a)$ is equal to 0. Hence the integers $d_1 = d_2 = 1$ satisfy the condition (1) of Definition 3.1 and since $|Y| = 2^m$, they satisfy also (2). The function g' is strongly hard.

For the function $1 - g'$ we set $J_1 = \{1\}$, $J_2 = \{2\}$, $Y = \{((a)_{+1}, a) \mid a \in \{0,1\}^m\}$. Since, for all $a \in \{0,1\}^m$, $(1 - g')(a, a) = 0$, both indices 1, 2 are important with respect to Y. Take any two vectors $((a)_{+1}, a), ((b)_{+1}, b) \in Y$ with $a \neq b$. We want to show that at least one of the values $(1 - g')((a)_{+1}, b)$, $(1 - g')((b)_{+1}, a)$ is equal to 0. Assume the contrary that both the values are equal to 1. Then we have $(a)_{+1} > b$ with $((a)_{+1}, b) \neq (1^m, 0^m)$ or $((a)_{+1}, b) = (0^m, 1^m)$. In the second case both a, b are equal to 1^m, which is a contradiction. In the first case we obtain $2^m - 1 > \mathrm{val}(a) + 1 > \mathrm{val}(b) > 0$ and similarly $2^m - 1 > \mathrm{val}(b) + 1 > \mathrm{val}(a) > 0$. These together imply the inequality $\mathrm{val}(a) + 1 > \mathrm{val}(b) > \mathrm{val}(a) - 1$. Thus again $a = b$, a contradiction. We have shown that at least one of values $(1 - g')((a)_{+1}, b)$, $(1 - g')((b)_{+1}, a)$ is equal to 0. The cardinality of the set Y is 2^m. Therefore as in the previous case the integers $d_1 = d_2 = 1$ satisfy both conditions (1) and (2). Hence, the function $1 - g'$ is strongly hard and the function g' is strongly very hard. $\square$

Using a similar method one can show the following lemma.

Lemma 3.3. *The function g is very hard.*

Now we show how to construct a very hard function $h(x_1, \ldots, x_n)$ using a strongly very hard function $f(x_1, x_2)$ and the operation *xor* on Boolean strings In order to do so let us define the operation xor on Boolean strings and the *xor constructor* $F^{\oplus}$ which constructs the function h.

Definition 3.4. Let $\oplus$ denote the Boolean operation *xor*. Let $x = x^1 \ldots x^m$, $y = y^1 \ldots y^m$ be m-bit strings. We define the xor operation on strings as follows: $x \oplus y = x^1 \oplus y^1 \ldots x^m \oplus y^m$. Let $n \geq 2$ be an integer, $J \subsetneq \{1, \ldots, n\}$ be a nonempty set and $f(x_1, x_2)$ be a Boolean function. We define the *xor constructor* $F^{\oplus}$ as follows

$$F^{\oplus}(J, n, f)(x_1, \ldots, x_n) = f\Big(\bigoplus_{i \in J} x_i, \bigoplus_{i \in \{1, \ldots, n\} - J} x_i \Big).$$

Theorem 3.5. *Let $f(x_1, x_2)$ be a strongly very hard function, $n \geq 2$ be an integer and $J \subsetneq \{1, \ldots, n\}$ be a nonempty set. Then the function $h = F^{\oplus}(J, n, f)$ is very hard.*

Proof. Since the function f is strongly hard, there exists a nonempty set $Y^f \subseteq f^{-1}(1)$ such that all indices are important with respect to Y^f and there are values of d_i^f such that

$$d_i^f \geq \max_{M \subseteq Y^f} \{|M| \mid C\ell_{J_i^f}(M) \subseteq f^{-1}(1)\} \quad \text{and} \quad \log \frac{|Y^f|}{d_i^f} = m,$$

where $J_i^f = \{i\}$, $i = 1, 2$. Denote $J' = \{1, \ldots, n\} - J$. Let us consider the following set

$$Y = \Big\{ (y_1, \ldots, y_n) \mid \Big(\bigoplus_{i \in J} y_i, \bigoplus_{i \in J'} y_i \Big) \in Y^f \Big\}.$$

First, we show that all indices are important with respect to Y for the function h. Let $j \in \{1, \ldots, n\}$ be an index and $(y_1, \ldots, y_n) \in Y$ be an input vector. Let

$$(x_1, x_2) = \Big(\bigoplus_{i \in J} y_i, \bigoplus_{i \in J'} y_i \Big) \in Y^f.$$

We can assume that $j \in J$, since the proof for $j \in J'$ is similar. Index 1 is important with respect to Y^f, therefore there is x_1' such that $f(x_1', x_2) = 0$. Put $y_j' = y_j \oplus x_1 \oplus x_1'$. We obtain

$$\Big(\bigoplus_{i \in J - \{j\}} y_i \Big) \oplus y_j' = \Big(\bigoplus_{i \in J} y_i \Big) \oplus x_1 \oplus x_1' = x_1'.$$

The value of xor on all y_i's with indices in the set J' remains constant, and hence $h(y_1, \ldots, y_{j-1}, y_j', y_{j+1}, \ldots, y_n) = f(x_1', x_2) = 0$. This implies that all indices are important with respect to Y.

We set $J_j = \{j\}$ for every $j \in \{1, \ldots, n\}$ and we prove that if we set $d_j = 2^{(n-2)m} \cdot d_1^f$ for all indices $j \in J$ and $d_j = 2^{(n-2)m} \cdot d_2^f$ for all indices $j \in J'$, then the assumptions of Theorem 2.4 are satisfied. To prove this it is enough to show for all indices $j \in J$ (for all indices $j \in J'$): if M is a subset of Y, which satisfies $C\ell_{J_j}(M) \subseteq h^{-1}(1)$ then $|M| \leq 2^{(n-2)m} \cdot d_1^f$ (then $|M| \leq 2^{(n-2)m} \cdot d_2^f$).

Hence take a $j \in J$ and a subset M of Y which satisfies $C\ell_{J_j}(M) \subseteq h^{-1}(1)$. (The proof is similar for $j \in J'$.) Let us define the following sets

$$\begin{aligned} K(x_1, x_2) &= \Big\{ (y_1, \ldots, y_n) \ \Big|\ (y_1, \ldots, y_n) \in M,\ \bigoplus_{i \in J} y_i = x_1,\ \bigoplus_{i \in J'} y_i = x_2 \Big\}, \\ L(x_1, x_2) &= \{x_1 \oplus y_j \mid (y_1, \ldots, y_n) \in K(x_1, x_2)\} \cdot \end{aligned}$$

Note that the sets $K(x_1, x_2)$ form a decomposition of the set M. For all $a \in \{0,1\}^m$ define the sets

$$M_f(a) = \{(x_1, x_2) \mid a \in L(x_1, x_2)\},$$

which form a decomposition of $[\{0,1\}^m]^2$. Note that $M_f(a) \subseteq Y^f$, since the sets $K(x_1, x_2)$ and $L(x_1, x_2)$ are empty if $(x_1, x_2) \notin Y^f$. We show that for any $a \in \{0,1\}^m$, $C\ell_{J_1^f}(M_f(a)) \subseteq f^{-1}(1)$, which implies that the set $M_f(a)$ has at most d_1^f elements.

Take two elements $x, x' \in M_f(a)$, where $x = (x_1, x_2)$, $x' = (x_1', x_2')$. We have $a \in L(x_1, x_2)$, so there exists a vector $y = (y_1, \ldots, y_n)$ which satisfies $y \in K(x_1, x_2)$ and $x_1 \oplus y_j = a$. Similarly there exists a vector $y' = (y_1', \ldots, y_n')$ for the values x_1', x_2'. Since both y, y' belong to the set M and $C\ell_{J_j}(M) \subseteq h^{-1}(1)$, we obtain $h([y : y'] / J_j) = 1$. Thus

$$\begin{aligned} f([x : x'] / J_1^f) &= f(x_1, x_2') = f(a \oplus y_j, x_2') = f(x_1' \oplus y_j' \oplus y_j, x_2') \\ &= f\Big(\Big(\bigoplus_{i \in J - \{j\}} y_i'\Big) \oplus y_j,\ \bigoplus_{i \in J'} y_i'\Big) = h([y : y'] / J_j) = 1. \end{aligned}$$

Therefore $C\ell_{J_1^f}(M_f(a)) \subseteq f^{-1}(1)$, which implies $|M_f(a)| \leq d_1^f$. We have

$$\ell = \sum_{(x_1, x_2)} |L(x_1, x_2)| = \sum_{a \in \{0,1\}^m} |M_f(a)| \leq 2^m \cdot d_1^f.$$

There is at most $2^{(n-3)m}$ elements in any $K(x_1, x_2)$ with the same value of y_j. So by the pigeon hole principle there is at least $\frac{|K(x_1,x_2)|}{2^{(n-3)m}}$ elements in any $K(x_1, x_2)$ with distinct values of y_j. Obviously such elements have also distinct values of $x_1 \oplus y_j$. Therefore we have the inequality $|L(x_1, x_2)| \geq \frac{|K(x_1,x_2)|}{2^{(n-3)m}}$. We can conclude

$$2^m \cdot d_1^f \geq \ell = \sum_{(x_1, x_2)} |L(x_1, x_2)| \geq \sum_{(x_1, x_2)} \frac{|K(x_1, x_2)|}{2^{(n-3)m}} = \frac{|M|}{2^{(n-3)m}} \cdot$$

This implies the inequality $|M| \leq 2^{(n-2)m} \cdot d_1^f$, as desired. So we can set $d_j = 2^{(n-2)m} \cdot d_1^f$ for all indices $j \in J$ and $d_j = 2^{(n-2)m} \cdot d_2^f$ for all indices $j \in J'$.

The cardinality of the set Y is $2^{(n-2)m} \cdot |Y^f|$. The rate between $|Y^f|$ and d_1^f is the same as the rate between $|Y|$ and d_i for any index $i \in J$ and similarly for d_2^f and any index $i \in J'$. It is easy to see that $\log \frac{|Y|}{d_i} = \log \frac{|Y^f|}{d_1^f} = m$ for all indices $i \in J$ and $\log \frac{|Y|}{d_i} = \log \frac{|Y^f|}{d_2^f} = m$ for all indices $i \in J'$. Hence, by Theorem 2.4, the communication complexity of the function h is nm, *i.e.*, function h is hard.

Since the complementary function $1 - f$ is also strongly hard and $1 - h = F^{\oplus}(J, n, 1 - f)$, the function $1 - h$ is hard too. This completes the proof. □

Theorem 3.5 gives us the method how to construct a very hard function for any n using a strongly very hard function $f(x_1, x_2)$. Hence, as a consequence of Theorems 3.2 and 3.5 we have:

Corollary 3.6. *Let $n \geq 2$ be an integer and J be a nonempty proper subset of $\{1, \ldots, n\}$. The function $F^{\oplus}(J, n, g')$ is very hard.*

Next, we analyze the functions created by combining two Boolean functions by the Boolean operation xor. More precisely:

Definition 3.7. Let $g_1 : [\{0,1\}^m]^{n_1} \to \{0,1\}$, $g_2 : [\{0,1\}^m]^{n_2} \to \{0,1\}$ be two Boolean functions. Let $n = n_1 + n_2$. We define the Boolean function $g_1 \oplus g_2$ as $\forall x \in [\{0,1\}^m]^n : (g_1 \oplus g_2)(x_1, \ldots, x_n) = g_1(x_1, \ldots, x_{n_1}) \oplus g_2(x_{n_1+1}, \ldots, x_n)$.

Now we can state our last theorem.

Theorem 3.8. *For all Boolean functions g_1, g_2*

$$\max(\mathrm{ncc}(g_1) + \mathrm{ncc}(1 - g_2), \mathrm{ncc}(1 - g_1) + \mathrm{ncc}(g_2)) \leq \mathrm{ncc}(g_1 \oplus g_2).$$

In particular, if both functions g_1 and g_2 are very hard, then the function $g_1 \oplus g_2$ is also very hard.

Proof. Let P be a protocol computing function $g_1 \oplus g_2$ with the communication complexity $\mathrm{ncc}(g_1 \oplus g_2)$. Let $J_1 = \{1, \ldots, n_1\}$ and $J_2 = \{n_1 + 1, \ldots, n\}$. Take an arbitrary $x = (x_1, \ldots, x_{n_1})$ such that $g_1(x) = 1$. We show that there exists $y_x = (x_{n_1+1}, \ldots, x_n)$ such that $g_2(y_x) = 0$ and $\mathrm{ncc}(P, (x, y_x)/J_2) \geq \mathrm{ncc}(1 - g_2)$. For the contrary assume that there is an x such that $g_1(x) = 1$ and for all $y \in g_2^{-1}(0)$ the protocol P needs to communicate on the input (x, y) less then $\mathrm{ncc}(1 - g_2)$ bits on the links in J_2. Now imagine the following protocol for function $1 - g_2$: the coordinator contains also the virtual parties $1, \ldots, n_1$; it simulates protocol P with the input x in the virtual parties. Clearly, since $g_1(x) = 1$ the protocol computes function $1 - g_2$. The communication complexity of the protocol is equal to the communication complexity of P on the links in J_2, since the communication between the coordinator and the virtual parties is performed inside the coordinator. But this is a contradiction, because by the assumption the protocol computes $1 - g_2$ with a smaller communication complexity than $\mathrm{ncc}(1 - g_2)$.

Now we show that there exists x_0 such that $\mathrm{ncc}(P,(x_0,y_{x_0})/J_1) \geq \mathrm{ncc}(g_1)$. Assume the contrary. Consider the following protocol for function g_1: the coordinator contains in addition the virtual parties $n_1+1,\ldots,n$; in the beginning it generates an arbitrary input $y \in g_2^{-1}(0)$ for the virtual parties and then it simulates protocol P. Clearly, such protocol computes the function g_1. Moreover, for each input x there exists a computation, when in the beginning the coordinator generated the input y_x for the virtual parties, and so this computation has the communication complexity less than $\mathrm{ncc}(g_1)$. Hence, the protocol has the communication complexity less than $\mathrm{ncc}(g_1)$, which is again a contradiction.

The existence of x_0 implies the inequality $\mathrm{ncc}(P) \geq \mathrm{ncc}(P,(x_0,y_{x_0})) \geq \mathrm{ncc}(g_1)+\mathrm{ncc}(1-g_2)$. By symmetry we have also $\mathrm{ncc}(P) \geq \mathrm{ncc}(g_2)+\mathrm{ncc}(1-g_1)$. These two inequalities give the result. □

As a consequence of Theorem 3.8 and Lemma 3.3 we have an another very hard function.

Corollary 3.9. *For all even n the function*

$$h\,(x_1,\ldots,x_n) = (\mathrm{val}(x_1) \leq \mathrm{val}(x_2)) \oplus \ldots \oplus (\mathrm{val}(x_{n-1}) \leq \mathrm{val}(x_n))$$

is very hard.

I would like to thank Pavol Ďuriš for many useful discussions.

References

[1] L. Babai, N. Nisan and M. Szegedy, *Multiparty protocols and logspace-hard pseudorandom sequences*, Proceedings, 21st ACM STOC (1989).

[2] N. Blum, A Boolean function requiring $3n$ network size. *Theoret. Comput. Sci.* **28** (1984) 337–345.

[3] A.K. Chandra, M.L. Furst and R.J. Lipton, *Multi-party protocols*, Proceedings, 15th ACM STOC (1983) 94–99.

[4] D. Dolev and T. Feder, *Multiparty communication complexity*, Proceedings, 30th IEEE FOCS (1989) 428–433.

[5] D. Dolev and T. Feder, Determinism *vs.* nondeterminism in multiparty communication complexity. *SIAM J. Comput.* **21** (1992) 889–893.

[6] P. Ďuriš and J.D.P. Rolim, A lower bound on the multiparty communication complexity, STACS'95. Springer–Verlag, *Lecture Notes in Comput. Sci.* **900** (1995) 350-360.

[7] P. Ďuriš, Multiparty communication complexity and very hard functions (to appear).

[8] J. Hromkovič, *Communication complexity and parallel computing*, An EATCS Series. Springer (1997).

[9] E. Kushilevitz and N. Nisan, *Communication complexity.* Cambridge Univ. Press. xiii (1997).

[10] R.J. Lipton and R. Sedgewick, *Lower bounds for VLSI*, Proceedings, 13th ACM STOC (1981) 300–307.

[11] O.B. Lupanov, Ob odnom metode sinteza skhem (Russian). *Izv. Vyssh. Uchebn. Zaved., Radiofizika* **1** (1958) 120–140.

[12] C. Shannon, The synthesis of two-terminal switching circuits. *Bell Syst. Techn. J.* **28** (1949) 59–98.

[13] A.C. Yao, *The entropic limitations on VLSI computations*, Proceedings, 13th ACM STOC (1981) 308–311.

Communicated by M.J. Hromkovic.
Received September 4, 1999. Accepted March 20, 2000.