ACADEMIC PRESS

Available at www.ComputerScienceWeb.com POWERED BY SCIENCE DIRECT®

Information and Computation 186 (2003) 78–89

Information and Computation

www.elsevier.com/locate/ic

A lower bound for integer multiplication on randomized ordered read-once branching programs☆

Farid Ablayev[a,*] and Marek Karpinski[b]

[a] *Max-Planck Institute for Mathematics, Bonn, Germany*[1]
[b] *Department of Computer Science, University of Bonn, Germany*

Received 18 February 1998; revised 14 February 2002

Abstract

We prove an exponential lower bound $2^{\Omega(n/\log n)}$ on the size of any *randomized* ordered read-once branching program computing *integer multiplication*. Our proof depends on proving a new lower bound on Yao's randomized one-way communication complexity of certain Boolean functions. It generalizes to some other models of randomized branching programs. In contrast, we prove that *testing integer multiplication*, contrary even to a nondeterministic situation, can be computed by *randomized* ordered read-once branching program in polynomial size. It is also known that computing the latter problem with deterministic read-once branching programs is as hard as factoring integers.

Keywords: Deterministic and randomized branching programs; OBDD; Complexity; Lower bounds; Integer multiplication; Randomized algorithms

1. Preliminaries

Oblivious (or *ordered*) read-once branching programs become an important tool in the field of digital design and verification (see, for example, [7,17]). They are also known under the name "OBDDs"

☆ Research partially supported by International Computer Science Institute, Berkeley, by DFG Grant KA 673/4-1 and by ESPRIT BR Grants 7097, and EC-US 030, and by the Max-Planck Research Prize.

* Corresponding author. Present address: Rheinische Friedrich-Wilhelms-Universität Bonn, Institut für Informatik, Römerstraÿe 164, Bonn DE-53117, Germany. Fax: +49-228-734440.

E-mail addresses: ablayev@ksu.ru (F. Ablayev), marek@cs.uni-bonn.de (M. Karpinski).

[1] Visiting from University of Kazan.

0890-5401/$ - see front matter © 2003 Elsevier Science (USA). All rights reserved.
doi:10.1016/S0890-5401(03)00118-4

(ordered binary decision diagrams). There are some important functions which are computationally *hard* for the OBDDs. One of such functions is the integer multiplication [6]. The other function is testing multiplication for which there is an exponential lower bound $2^{\Omega(n^{1/4})}$ known even for the nondeterministic OBDDs [10]. An interesting open problem remained whether randomization can help in the computation of these functions by the OBDDs. In this paper we show, for the first time, that the method of [3] yields a polynomial size ($O(n^6 \log^4 n)$) bound for the latter function on randomized OBDDs. Interestingly, it is known that computing that function by the deterministic read-once branching programs is as hard as the integer factoring [13,17]. Further, we prove an exponential lower bound $2^{\Omega(n/\log n)}$ on the size of any randomized OBDD computing the *integer multiplication*.

During the last decade there were several attempts to find appropriate generalizations of a OBDD model for hardware verification, strong enough to compute efficiently integer multiplication. But again, the results showed that the integer multiplication remained hard also for these models [9,13].

In [3], a randomized model of a branching program was introduced. The usefulness of that model was highlighted by the fact that there are many interesting functions which are hard for deterministic OBDDs but are easy for randomized OBDDs. The first such a function was discovered in [3]. Among these functions is also a *clique-only function* which is hard even for more general model of nondeterministic syntactic read-k-times branching programs [4] (see also [18] for more examples).

It was proved in [2] that the randomized and nondeterministic models of OBDD are incomparable. There was still a hope (note that the multiplication is hard for nondeterministic OBDDs [9]) that randomized OBDDs can compute the integer multiplication in polynomial size. Our results show that randomized OBDDs can test integer multiplication in polynomial size but the integer multiplication itself requires exponential size.

Up to now it was not clear what is harder to multiply or to test the multiplication (see [14] for more information). It is known that *DMULT* (testing multiplication) is hard for the *syntactic nondeterministic read-k-times* branching programs [10]. Note also that *DMULT* function is AC^0 equivalent to *MULT* [8]. Our result answers thus to the problem raised in [17] about succinct representations of the functions *DMULT* and *MULT*.

2. Basic definitions and results

We recall now some basic definitions (cf. [15,18]).

A *deterministic* branching program P for computing a Boolean function $g : \{0,1\}^n \to \{0,1\}$ is a directed acyclic multi-graph with a distinguished source node s and a distinguished sink node t. The outdegree of each nonsink node is exactly 2 and the two outgoing edges are labeled by $x_i = 0$ and $x_i = 1$ for the variable x_i associated with this node. We call such a node an x_i-node. The label "$x_i = \delta$" indicates that only the inputs satisfying $x_i = \delta$ may follow that edge in the computation. The branching program P computes a function g in an obvious way: for each $\sigma \in \{0,1\}^n$ we let $f(\sigma) = 1$ iff there is a directed s–t path starting in the source s and leading to the (accepting) node t such that all labels $x_i = \sigma_i$ along this path are consistent with $\sigma = \sigma_1\sigma_2 \cdots \sigma_n$.

We define a *randomized* branching program [3] as a branching program having in addition specially designated *random* ("coin-toss") inputs. The values of these random inputs are chosen from a uniform distribution, and an output of a randomized branching program become a random variable.

We say that a randomized branching program (a, b)-computes a Boolean function g if it outputs 1 with probability at most a for input σ such that $g(\sigma) = 0$ and outputs 1 with probability at least b for inputs σ such that $g(\sigma) = 1$. For $1/2 < p \leqslant 1$ we write shortly "p-computes" instead of "$(1 - p, p)$-computes." A randomized branching program computes a function g with a one-sided ε-error if g is $(\varepsilon, 1)$-computed. We define the size of P, $size(P)$ as the number of its *internal* nodes (we refer to it sometimes as the complexity of P).

A read-once branching program is a branching program in which every variable is tested at most once in every path. A τ-ordered read-once branching program is a read-once branching program which respects an ordering τ of the variables, i.e., if an edge leads from an x_i-node to an x_j-node, the condition $\tau(i) < \tau(j)$ has to be fulfilled. An OBDD (alternatively, ordered read-once branching program) is a τ-ordered read-once branching program respecting some ordering τ of variables.

In the rest of this section we present main results of the paper. We start with defining a Boolean decision function: *the testing integer multiplication function* (or alternatively, *decision problem of recognizing the graph of multiplication*) *DMULT* as follows. $DMULT : \{0, 1\}^{4n} \to \{0, 1\}$ and $DMULT(X, Y, Z) = 1$ iff $XY = Z$. Here X, Y, and Z are binary representations of integer numbers, $|X| = |Y| = n$, $|Z| = 2n$.

Theorem 1. *Function DMULT can be computed by a randomized OBDD with a one-sided $\varepsilon(n)$-error of size*

$$\mathrm{O}\left(\frac{n^6}{\varepsilon^5(n)}\log^4\frac{n}{\varepsilon(n)}\right).$$

We define now an *integer multiplication function MULT* as follows. The function $MULT_k : \{0, 1\}^{2n} \to \{0, 1\}$ defines the kth bit, $0 \leqslant k \leqslant 2n - 1$ in the product of two n-bit integers. That is, $MULT_k(X, Y) = z_k$, where $X = x_{n-1} \cdots x_0$, $Y = y_{n-1} \cdots y_0$, and $Z = z_{2n-1} \cdots z_0$. Now denote by *MULT* the function $MULT_n$ which computes the *middle* bit in the product xy. It is known that the middle bit is the "hardest" bit (see, for example [13]).

For $p \in (1/2, 1)$, $k \in \{0, \ldots, 2n - 1\}$, and a permutation τ of $\{1, \ldots, 2n\}$ let $P_p(k, \tau)$ be a randomized OBDD with an ordering τ that p-computes $MULT_k$.

Theorem 2. *Given $p \in (1/2, 1)$. For every τ there exists a k such that*

$$size(P_p(k, \tau)) \geqslant 2^{n(1-H(p))/8},$$

where $H(p) = -p \log p - (1 - p)\log(1 - p)$ is the Shannon entropy.

Theorem 3. *Let, for $p \in (1/2, 1)$, the function $MULT(X, Y)$ be p-computed by a randomized OBDD P. Then*

$$size(P) \geqslant 2^{\Omega(n/\log n)}.$$

The above theorems state that multiplication is in fact hard for randomized OBDDs. The first theorem is "weaker" than the second one. However the proof of the first theorem is shorter and more direct. It is based on a proof of a lower bound for the polynomial projection function (subfunction) [5]. The proof of the Theorem 3 itself is based on a lower bound for another polynomial projection of *MULT* [6,9] using randomized binary search communication game. Proofs of the theorems are presented in the next section.

3. Randomized OBDDs for testing multiplication

In this section we present a proof of Theorem 1. Let $d(n)$ be some function in $O(n)$ such that $d(n) > 4n$.

Lemma 1. *Let $\varepsilon(n) < 4n/d(n)$. Then DMULT can be computed by a randomized OBDD with one-sided $\varepsilon(n)$-error of size*

$$O(nd(n)^5 \log^4 d(n)).$$

Proof. The following randomized (fingerprinting) algorithm tests the multiplication. Uniformly at random select a prime number p from the set $Q_{d(n)} = \{p_1, \ldots, p_{d(n)}\}$ of the first $d(n)$ primes. Then deterministically compute $a = X \bmod p$, $b = Y \bmod p$, multiply ab, then compute $c = Z \bmod p$, and verify whether $ab = c$. If $ab = c$ then *accept* else *reject*. Chinese reminder theorem provides the correctness of such a computation and the fingerprint arguments of [3] provide a correct result for testing $XY = Z \bmod p$ by randomized OBDDs with high probability. All these manipulations can be done by a polynomial size randomized OBDD P constructed below.

Phase 1. (Randomized). P randomly selects a prime number p from the set $Q_{d(n)} = \{p_1, p_2, \ldots, p_{d(n)}\}$ of the first $d(n)$ prime numbers.

P uses $t = \lceil \log d(n) \rceil$ random bits for selecting a prime number p. P reads random bits in the order $\xi_1, \ldots, \xi_t$. We view $\xi = \xi_1 \ldots \xi_t$ as a binary representation of an integer. P selects ith prime number $p_i \in Q_{d(n)}$ iff $\xi = i \bmod d(n)$.

Phase 2. (Deterministic). Along its computation path P computes $a = X \bmod p$, by reading consecutive bits from X. P stores a in an internal node (state). Then, P computes $b = Y \bmod p$ and stores the product ab. At last P counts $c = Z \bmod p$ and verify whether $ab = c$. If $ab = c$ then it *accepts* else it *rejects*.

So, if $XY = Z$, then P with probability 1 outputs the correct answer. If $XY \neq Z$, then it can happen that $XY = Z \pmod p$ for some $p \in Q_{d(n)}$. In these cases P makes an error.

For $XY \neq Z$ we have $|XY - Z| \leqslant 2^{2n} < p_1 \cdots p_{2n}$, where $p_1, \ldots, p_{2n}$ are the first $2n$ prime numbers. This means that in the case when $XY \neq Z$, the probability $\varepsilon(n)$ of the error of P on the input X, Y, Z is less than or equal to $4n/d(n)$ (less than or equal to $2n/d(n)$ if t is a power of 2).

For $p \in Q_{d(n)}$ denote by S_p a deterministic subprogram of P that carries out the deterministic part of computations of the *phase 2* with the prime p.

The size of P is bounded by

$$2^{t+1} - 1 + \sum_{p \in Q_{d(n)}} size(S_p).$$

S_p is a deterministic OBDD that realizes the phase 2 of the algorithm above. S_p reads variables in the order X, Y, Z and therefore has the length $4n$. For the realization of the procedure described in the *phase 2* it is sufficient to store in the internal nodes (in the internal states of S_p) four numbers: $X \bmod p$, $Y \bmod p$, $XY \bmod p$, and $Z \bmod p$. The ith prime is of order $O(i \log i)$. Therefore we have

$$size(S_p) = O(np^4) = O(n(d(n) \log d(n))^4). \qquad \square$$

Our Lemma entails now Theorem 1.

4. Lower bounds

For proving lower bounds we use Yao's standard randomized communication complexity model [19,20] (see also [11,12]) for Boolean functions.

We recall basic definitions of a one-way communication complexity model. Consider a Boolean function $g : \{0,1\}^n \to \{0,1\}$. Let $\pi = (L, R)$ be a partition of a set of variables of g into two parts. The first argument, L, of g is known to the first player I, and the second argument, R, is known to the second player II. Player I starts computation on its part of an input. Player II on obtaining a message (binary string) from I and its part of an input produces a result. The number of bits in the message is the communication complexity of a specific communication protocol. The communication complexity of the function g is the communication complexity of the best protocol for g.

Let $\varepsilon \in (0, 1/2)$, $p = 1/2 + \varepsilon$. A randomized communication protocol Φ p-computes a function g for a partition π of inputs if for every input (σ, γ) of g it holds that $Pr(\Phi(\sigma, \gamma) = g(\sigma, \gamma)) \geqslant p$. Note that a notion of a p-computation corresponds to a computation with δ-error for $\delta = 1 - p$, cf., e.g. [12]. Denote by $PC_p^{\pi}(g)$ a randomized one-way p-communication complexity of g according to the partition π of inputs.

The following lemma is proved in [2]. It gives a connection between a size of a OBDD and its one-way communication complexity.

Lemma 2 (cf. [2])**.** *Let $\varepsilon \in (0, 1/2)$, and $p = 1/2 + \varepsilon$. Let a randomized OBDD P p-compute g. Let $\pi = (L, R)$ be a partition of inputs between players with L and R defined according to an ordering τ of inputs of P. That is, P can read variables from R only after reading variables from L and cannot read variables from L after starting reading variables from R. Then*

$$size(P) \geqslant 2^{PC_p^{\pi}(g)-1}.$$

Denote by CM^{π} a communication matrix for Boolean function g for a partition π of inputs of g. In this paper we consider only functions with the property that all rows of their communication matrixes are different.

Choose a set $Z \subseteq R$ such that for an arbitrary two settings σ, σ' of variables from L there exists a setting γ of variables from Z such that $g(\sigma, \gamma) \neq g(\sigma', \gamma)$. *The set Z is called the control set for the matrix CM^{π}*. Denote by $ts(CM^{\pi})$ the minimum size of a control set for the matrix CM^{π} and $nrow(CM^{\pi})$ the number of different rows of the matrix CM^{π}.

For $p = 1/2 + \varepsilon$, define a probabilistic communication characteristic $pcc_p^{\pi}(g)$ of g (see [1]) as follows:

$$pcc_p^{\pi}(h) = \frac{ts(CM^{\pi})}{\log nrow(CM^{\pi})} H(p),$$

where $H(p) = -p \log p - (1-p)\log(1-p)$. The following theorem [1] states that the randomized one-way communication complexity cannot be too "small" for a function with a "large" *data set* and a "small" *control set*.

Theorem 4 (cf. [1])**.** *Let $\varepsilon \in [0, 1/2]$, and $p = 1/2 + \varepsilon$. Let $\pi = (L, R)$ be a partition of a set of inputs of g. Then*

$$PC_p^{\pi}(g) \geqslant DC^{\pi}(g)(1 - pcc_p^{\pi}(g)) - 1,$$

where $DC^{\pi}(g)$ is the deterministic one-way communication complexity of g for a partition π of inputs.

4.1. Proof of Theorem 2

Our proof proceeds as follows:

(i) we construct a polynomial projection f^k of $MULT_k$ and then

(ii) we prove that f^k is hard for a randomized τ-ordered OBDD.

For an ordering τ of the variables of $P_p(k, \tau)$, there are two subsets L and R of equal size $l \geqslant n/2$ such that:

(1) $P_p(k, \tau)$ reads all variables from L before starting reading variables from R and

(2) $L \subset X$ and $R \subset Y$ or $L \subset Y$ and $R \subset X$.

W.l.g. assume in the rest of the proof that $L \subset X$ and $R \subset Y$. Thus, $L = \{x_{i_1}, \ldots, x_{i_l}\}$ and $R = \{y_{j_1}, \ldots, y_{j_l}\}$.

We will be interested now only in the inputs $\sigma \in \{0, 1\}^{2n}$ such that: for the variables Y, all bits in σ except for a one bit of R are 0. Call such an R a *control* set. Variables from L can take arbitrary values from $\{0, 1\}$. For convenience also fix the remaining variables from $X \backslash L$ to be 0. Call such an L a *data* set.

For an integer m, $1 \leqslant m \leqslant 2n$, denote by $[m]$ a set of pair of bits of the data and control sets that are transmitted to the mth bit of the product XY. Formally

$$[m] = \{(x_i, y_j) \in L \times R : i + j = m\}.$$

Since $|L \times R| = l^2 \geqslant n^2/4$ there exists an integer k such that

$$|[k]| = t \geqslant l^2/(2n) = n/8. \tag{1}$$

Now fix this integer k. Denote by $L_k \subset L$ ($R_k \subset R$) a subset of L (R) that consists of all variables x_i (y_j) that "take part" in the set $[k]$.

Consider a projection $f^k : L_k \times R_k \to \{0, 1\}$ of $MULT_k$, for which all variables from $(Y \cup X) \backslash (L_k \cup R_k)$ are fixed to 0. The communication matrix CM^{π} of f^k for a partition $\pi = (L_k, R_k)$ of inputs has the following property: CM^{π} is $2^t \times t$ Boolean matrix and all rows of CM^{π} are different. From that we have that $pcc_p^{\pi}(f^k) = H(p)$ and $DC^{\pi}(f^k) = \log t$. From the above we get that

$$size(P_p(k, \tau)) \geqslant 2^{t(1-H(p))}.$$

Using (1) and the inequality above we get the lower bound of the Theorem.

4.2. Proof of Theorem 3

The proof consists of 3 steps:

(i) we construct a polynomial projection SUM of $MULT$ (see [6,9]) for SUM a Boolean function that computes the most significant bit of the sum of two integers,

(ii) using a randomized OBDD P for $MULT$ (which turnes out to be a randomized OBDD for SUM with proper value assignments to the variables) construct a randomized one-way communication protocol for computing the function SUM, and

(iii) finally, we prove a lower bound of the theorem, using the facts
- that the randomized one-way communication complexity gives a lower bound for the randomized OBDD size and
- that SUM is hard for randomized one-way communication computation.

For simplification of the technical details of the proof we assume that n is an even number. Let τ be an ordering of variables of a randomized OBDD P. Denote by X_1, $|X_1| = n/2$, (X_2, $|X_2| = n/2$) the first half (the remaining part) of the set X of variables tested (in the order τ) by P. Denote by U and W the two subsets of X_1 and X_2, respectively, such that: either all indices of the variables from U are smaller than the indices of the variables from W, or vice versa, all indices of the variables from W are smaller than the indices of the variables from U. The following lemma shows that we can choose such sets U, W to be large enough.

Lemma 3. *There exist sets U and W such that*

$$n/4 \leqslant |U| = |W| \leqslant n/2.$$

Proof. Denote by m_1 and M_1 the minimum and the maximum value of the indices of the variables from X_1. Denote by m_2 and M_2 the minimum and the maximum value of the indices of variables from X_2. If $M_1 < m_2$ or $M_2 < m_1$ then set $U = X_1$, $W = X_2$, and we are done.

Now consider the remaining case $m_1 < M_2$ and $m_2 < M_1$. The following algorithm constructs U and W as needed.

Begin{procedure $\mathcal{A}$}

Step 1: Put $Z_1 = \{x_{m_1}\}$ and $Z_2 = \{x_{M_2}\}$.

Step i: ($2 \leqslant i < n/2$). For $R \subseteq X$ denote by $I(R)$ a set of indexes of variables from R. Let $x_m \in X_1$ and $x_M \in X_2$. Call the pair of variables (x_m, x_M) a (Z_1, Z_2)-good pair if $m = \min\{i \in I(X_1 \backslash Z_1)\}$, $M = \max\{i \in I(X_2 \backslash Z_2)\}$ and $m < M$.

if there exists a (Z_1, Z_2)-good pair **then** add x_m to Z_1 and add x_M to Z_2 **else** stop $\mathcal{A}$.

End{procedure $\mathcal{A}$}

Put $Z_1' = X_1 \backslash Z_1$ and $Z_2' = X_2 \backslash Z_2$. We clearly have $|Z_1| = |Z_2|$ and $|Z_1'| = |Z_2'|$. From the description of the procedure $\mathcal{A}$ it follows that all indices of the variables from Z_1 are smaller than the indices of the variables from Z_2 and all indices of the variables from Z_2' are smaller than the indices of the variables from Z_1'. Let $a = |Z_1|$.

If $a \geqslant n/4$ **then** put $U = Z_1$ and $W = Z_2$ **else** put $U = Z_1'$ and $W = Z_2'$. □

Now fix the sets U and W satisfying Lemma 3. Without loss of generality we assume that all indices of the variables in U are smaller than the indices of the variables in W.

Lemma 4. *There exist an integer k, $1 \leqslant k \leqslant n$, and sets $L \subseteq U$, $R \subseteq W$ such that $|L| = |R| = l \geqslant n/16$ and $(L, R) = \{(x_i, x_j) : x_i \in L,\ x_j \in R,\ j = i + k\}$.*

Proof. Let $V = U \times W$. For an integer $k \in \{1, \ldots, n\}$ define a set $[k] = \{(x_i, x_j) \in V : j = i + k\}$. Clearly we have that $V = \cup_{k=1}^{n}[k]$ and $[k] \cap [k'] = \emptyset$ for $k \neq k'$. Since $|V| \geqslant n^2/16$ for some $k \in \{1, \ldots, n\}$, we have that $|[k]| \geqslant n/16$. □

Now fix the sets $L = \{x_{i_{l-1}}, \ldots, x_{i_0}\}$ and $R = \{x_{j_{l-1}}, \ldots, x_{j_0}\}$ satisfying Lemma 4. View sequences L and R as binary representation of numbers. Define now the Boolean function $SUM(L, R)$ as follows. $SUM(\sigma, \gamma) = 1$ iff $\sigma + \gamma \geqslant 2^l$. That is, for $\sigma = \sigma_{l-1}, \ldots \sigma_0$ and $\gamma = \gamma_{l-1}, \ldots, \gamma_0$
$SUM(\sigma_{l-1}, \ldots \sigma_0, \gamma_{l-1}, \ldots, \gamma_0) = 1$ iff $\sum_{i=0}^{l-1}(\sigma_i + \gamma_i)2^i \geqslant 2^l$.

For a given Boolean function $f(X, Y)$, a subset $Z \subseteq X \cup Y$, and an assignment $\rho : (X \cup Y)\backslash Z \to \{0, 1\}$, denote by $f|_\rho(X, Y)$ a subfunction of $f(X, Y)$ with the variables from $(X \cup Y)\backslash Z$ fixed according to ρ.

The next lemma states a fact used also earlier in [9].

Lemma 5. *There exists an assignment ρ of variables from $(X \cup Y)\backslash(L \cup R)$ ($\rho : (X \cup Y)\backslash(L \cup R) \to \{0, 1\}$) such that* (1) $SUM(L, R) = MULT|_\rho(X, Y)$, (2) *for a partition $\pi = (L, R)$, a communication matrix CM^π for SUM has the following structure* (*for a suitable ordering of rows and columns*) : *all the elements on and above the second diagonal are* 0, *and all the elements below the second diagonal are* 1.

Proof. Denote by J a set of indexes of R. Let $m = \min\{i \in J\}$ and $M = \max\{i \in J\}$. Define now ρ as follows: for $x_j \in X\backslash(L \cup R)$

$$\rho(x_j) := \begin{cases} 1 & \text{if}(j \notin J) \bigwedge (m < j < M), \\ 0 & \text{otherwise,} \end{cases}$$

for $y_j \in Y$

$$\rho(y_j) := \begin{cases} 1 & \text{if } j = n - 1 - M, \\ 1 & \text{if } j = n - 1 - (M - k), \\ 0 & \text{otherwise.} \end{cases}$$

From the definition of ρ it follows that $SUM(L, R) = MULT|_\rho(X, Y)$. The rows and columns of CM^π are indexed by the integers $\sigma \in [0, 2^l - 1]$ and $\gamma \in [0, 2^l - 1]$. The (σ, γ) entry of CM^π is $SUM(\sigma, \gamma)$. We fix the ordering of rows and columns of CM^π according to the increasing order of their indexes. From this we have that all entries of CM^π on and above the diagonal $i + j = 2^l - 1$ are equal to 0 and all entries below the diagonal are equal to 1. □

We assume in the remaining part of the proof that the variables from $(Y \cup X)\backslash(L \cup R)$ have been fixed as needed. So P is turned to a randomized OBDD that p-computes $SUM(L, R)$.

Below, using P, we construct a randomized one-way communication protocol Φ for arbitrary Boolean function g. Then we apply this communication protocol for a particular "pointer" function g_{pt} (defined below) with a high one-way randomized communication complexity.

Let $g(L, Z)$ be an arbitrary Boolean function over a fixed set L of variables and a "new" set $Z = \{z_1 \ldots, z_k\}$ of variables, that is, $Z \cap (X \cup Y) = \emptyset$.

Lemma 6. *For $q \in (1/2, 1)$ there exists a randomized one-way communication protocol Φ for q-computing function $g(L, Z)$ such that*

$$C(\Phi) \leqslant a(\log b\, l)(\log size(P)),$$

where a, b are positive constants.

Proof. We describe a randomized one-way communication protocol Φ for q-computing g as follows. Let $\sigma = \sigma_1, \ldots, \sigma_l$ be an input sequence of player I and $\omega = \omega_1, \ldots, \omega_k$—an input sequence of player II. Let $t = a \log(bl)$. We define constants a, b in a proper way later. Players I and II use branching program P for their computations as follows. I runs branching program P on its part of inputs t times and sends t nodes $v_1, \ldots, v_t$ which were reached by P during the computations to the player II. Player II uses the branching program P and the communication matrix CM^π of $SUM(L, R)$. The goal of the player II is to determine the input string σ of the player I with probability no less than q (more precisely the player II determines a string σ' such that probability of the event $\sigma' = \sigma$ is no less than q). Then, the player II having its part of an input, outputs the correct result $g(\sigma, \omega)$ with probability no less than q. Let $B_0 := \{0, 1\}^l$. In each step $i \geqslant 1$, II reduces a set B_{i-1} and in the last step l of the procedure, II gets a set $B_l = \{\sigma'\}$. Player II after getting $v_1, \ldots, v_t$, determines σ' by a randomized binary search procedure as follows.

Step 1. Take a "middle" sequence γ^1 of the possible valuations of the subset R of the variables of $SUM(L, R)$. That is, a sequence γ^1 determines the middle column of the communication matrix CM^π.

Run P on γ^1 t times starting from nodes $v_1, \ldots, v_t$, and take the majority result $\Delta_1 \in \{accept, reject\}$. Using Δ_1, select a set B_1 of potential inputs of player I (the set of sequences that determine the upper half of rows of CM^π or the set of sequences that determine the lower half of rows of CM^π). $|B_1| = 2^l/2$.

Step 2. If $\Delta_1 = accept$ then select a "middle" input sequence γ^2 between γ^1 and $\mathbf{1} = (1, \ldots, 1)$ else—between $\mathbf{0} = (0, \ldots, 0)$ and γ^1.

Run P on γ^2 t times starting from nodes $v_1, \ldots, v_t$ and take the majority result $\Delta_2 \in \{0, 1\}$. Using Δ_2, select a set $B_2 \subset B_1$ of potential inputs of player I. $|B_2| = |B_1|/2$.

After l steps the procedure stops by selecting a set B_l that consists of the unique input sequence σ'. Player II outputs the result $g(\sigma', \omega)$. Clearly, we have

$$C(\Phi) \leqslant t \log size(P).$$

The following counting arguments show that protocol Φq-computes g.

For a string $\gamma^i \in \{0, 1\}^l$ that determines a column of the matrix CM^π we denote by $Pr(\gamma^i)$ a probability of getting the correct result Δ_i in the step i, $1 \leqslant i \leqslant l$, by the binary search procedure above. Then the probability $Pr(\sigma' = \sigma)$ of correctly determining an input of player I is

$$Pr(\sigma' = \sigma) = Pr(\gamma^1) \ldots Pr(\gamma^l).$$

The probability $1 - Pr(\gamma^i)$ of getting error Δ_i is no more than $(1/c(p))^t$ for some constant $c(p) > 1$ depending on the probability p of correct computation of P. By choosing a constant a in a proper way we get

$$1 - Pr(\gamma) \leqslant 1/(b\,l).$$

From the above it follows that

$$Pr(\sigma' = \sigma) \geqslant (1 - 1/(b\,l))^l.$$

Using the fact that function $(1 - 1/x)^{x/b}$ is monotonically increasing to $(1/e)^{1/b}$ for $x \to \infty$ we get for properly selected constant $b > 1$ and for l large enough

$$Pr(\sigma' = \sigma) \geqslant q. \qquad \square$$

Below we apply the communication protocol Φ from our Lemma 6 for computing a certain "pointer" function with a high one-way randomized communication complexity. Let $pt : \{0,1\}^n \to \{1,\ldots,n\}$, we will call such a function pt a *pointer*. A pointer Boolean function $f_{pt} : \{0,1\}^n \to \{0,1\}$ is defined as $f_{pt}(\sigma) = \sigma_j$, where $j = pt(\sigma)$. The following pointer function g_{pt} was firstly considered in [16]. A pointer pt is defined as follows: $j = \omega(\sum_{i=1}^{n} i\sigma_i)$ where $\omega(s)$ is determined as follows. For an integer n let $p[n]$ be the smallest prime number greater than or equal to n. Then, for every integer s, let j be the unique integer satisfying $j = s \bmod p[n]$ and $1 \leqslant j \leqslant p[n]$. Then, $\omega(s) = j$, if $1 \leqslant j \leqslant n$, and $\omega(s) = 1$ otherwise.

In its communication complexity variant, a pointer function $g_{pt}(L, Z)$ is defined as follows: for a valuations $\sigma = \sigma_1 \ldots, \sigma_l$ and $\sigma' = \sigma_{l+1}, \ldots, \sigma_n$ of variables $g_{pt}(\sigma, \sigma') = \sigma_j$ for $j = pt(\sigma, \sigma')$.

We formulate now our last lemma.

Lemma 7. *For arbitrary $q \in (1/2, 1)$, and arbitrary $\delta > 0$, and for every l large enough, we have*

$$PC_q(g) \geqslant (l - o(l))(1 - (1+\delta)H(q)),$$

where $H(q) = -q \log q - (1-q)\log(1-q)$ is the Shannon entropy.

See [2] for a proof of the above Lemma.

Finally from Lemmas 6, and 7, and using the fact that $n/16 \leqslant l \leqslant n/2$, we get the lower bound for the $size(P)$ stated in our theorem.

5. Generalization and concluding remarks

Note that our proof technique of the previous section has used the following essential fact. The set of variables of a program P can be partitioned (according to the ordering τ of P) into two parts L and R (of approximately equal size) such that for any computation path of P the following is true. If a variable from R is tested, then no variable from L can be tested in the rest of this path. This means that the statement of Theorem 3 remains also true for other natural models of branching programs we define below.

We define a *balanced partition* of a set X as any partition of a set X into subsets X_1 and X_2 satisfying $|X_1| = \Theta(|X_2|)$.

Definition 1. We call a branching program P a π-balanced-weak-ordered branching program if it respects a balanced partition π of its variables X into two parts X_1 and X_2 such that if an edge leads from an x_i-node to an x_j-node, where $x_i \in X_t$ and $x_j \in X_m$, then the condition $t \leqslant m$ has to be fulfilled.

Call a branching program P a balanced-weak-ordered if it is π-balanced-weak-ordered for some partition π of the set of variables of P into two sets.

Our Theorem 3 can be generalized as follows.

Theorem 5. *Let for $p \in (1/2, 1)$ the function $MULT(X, Y)$ be p-computed by a randomized balanced-weak-ordered branching program P. Then*

$$size(P) \geqslant 2^{\Omega(n/\log n)}.$$

Open problems. There remains an interesting open problem on the lower bounds for the integer multiplication on randomized branching programs with (1) the limited number of inputs readings and (2) without any condition on the ordering of variables. We conjecture that the corresponding lower bounds are also exponential.

Acknowledgments

We would like to thank Anna Gál, Stephen Ponzio, Sasha Razborov, Thomas Thierauf, and Andy Yao for helpful discussion on the subject of the paper.

References

[1] F. Ablayev, Lower bounds for one-way probabilistic communication complexity, in: Proceedings of the ICALP'93, Lecture Notes in Computer Science, vol. 700, Springer, Berlin, 1993, pp. 241–252.
[2] F. Ablayev, Randomization and nondeterminism are incomparable for ordered read-once branching programs, in: Proceedings of the ICALP'97, Lecture Notes in Computer Science, vol. 1256, Springer, Berlin, 1997, pp. 195–202.
[3] F. Ablayev, M. Karpinski, On the power of randomized ordered branching programs, Research Report 85181-CS, University of Bonn, 1997.
[4] A. Borodin, A. Razborov, R. Smolensky, On lower bounds for read-k-times branching programs, Comput. Complex. 3 (1993) 1–18.
[5] R. Bryant, Graph-based algorithms for boolean function manipulation, IEEE Trans. Comput. C-35 (8) (1986) 677–691.
[6] R. Bryant, On the complexity of VLSI implementations and graph representations of Boolean functions with applications to integer multiplication, IEEE Trans. Comput. 40 (2) (1991) 205–213.
[7] R. Bryant, Symbolic boolean manipulation with ordered binary decision diagrams, ACM Comput. Surv. 24 (3) (1992) 293–318.
[8] R. Buss, The graph of multiplication is equivalent to counting, Inf. Process. Lett. 41 (1992) 199–201.
[9] J. Gergov, Time-space tradeoffs for integer multiplication on various types of input oblivious sequential machines, Inf. Process. Lett. 51 (1994) 265–269.
[10] S. Jukna, The graph of integer multiplication is hard for read-k-times networks, TR 95-10 Mathematik/Informatik University of Trier, 1995.
[11] J. Hromkovic, Communication Complexity and Parallel Computing, EATCS Series, Springer, Berlin, 1997.
[12] E. Kushilevitz, N. Nisan, Communication Complexity, Cambridge University Press, 1997.
[13] S. Ponzio, A lower bound for integer multiplication with read-once branching programs, in: Proceedings of the 27-th STOC, 1995, pp. 130–139.
[14] S. Ponzio, Restricted branching programs and hardware verification, Technical Report, MIT/LCS-TR-633, MIT, 1995.
[15] A. Razborov, Lower bounds for deterministic and nondeterministic branching programs, in: Proceedings of the FCT'91, Lecture Notes in Computer Science, vol. 529, Springer, Berlin, 1991, pp. 47–60.
[16] P. Savicky, S. Zak, A large lower bound for 1-branching programs, Electronic Colloquium on Computational Complexity, Revision 01 of TR96-036 (1996). Available From http://www.eccc.uni-trier.de/eccc/.

[17] I. Wegener, Efficient data structure for Boolean functions, Discrete Math. 136 (1994) 347–372.
[18] I. Wegener, Branching programs and binary decision diagrams. Theory and applications, SIAM Monographs on Discrete Mathematics and Applications, 2000.
[19] A.C. Yao, Some complexity questions related to distributive computing, in: Proceedings of the 11th Annual ACM Symposium on the Theory of Computing 1979, pp. 209–213.
[20] A.C. Yao, Lower bounds by probabilistic arguments, in: Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science, 1983, pp. 420–428.

Further reading

[1] F. Ablayev, M. Karpinski, On the power of randomized branching programs, in: Proceedings of the ICALP'96, Lecture Notes in Computer Science, vol. 1099, Springer, Berlin, 1996, pp. 348–356.