JOURNAL OF COMPUTER AND SYSTEM SCIENCES 45, 204–232 (1992)

Multiparty Protocols, Pseudorandom Generators for Logspace, and Time–Space Trade-offs

LÁSZLÓ BABAI*

University of Chicago, Chicago, Illinois 60637-1504, and Eötvös University, Budapest, Hungary

NOAM NISAN†

Hebrew University, Mount Scopus, Jerusalem, Israel

AND

MÁRIÓ SZEGEDY*

AT&T Bell Laboratories, Murray Hill, New Jersey 07974-0636

Received August 17, 1989

Let $f(x_1, ..., x_k)$ be a Boolean function that k parties wish to collaboratively evaluate, where each x_i is a bit-string of length n. The ith party knows each input argument except x_i; and each party has unlimited computational power. They share a blackboard, viewed by all parties, where they can exchange messages. The objective is to minimize the number of bits written on the board. We prove lower bounds of the form $\Omega(n \cdot c^{-k})$, for the number of bits that need to be exchanged in order to compute some (explicitly given) polynomial time computable functions. Our bounds hold even if the parties only wish to have a 1% advantage at guessing the value of f on random inputs. The lower bound proofs are based on discrepancy upper bounds for specific functions over "cylinder intersection" sets. These results may be of independent interest. We give several applications of the lower bounds. The first application is a pseudorandom generator for Logspace. We explicitly construct (in polynomial time) pseudorandom sequences of length n from a random seed of length $\exp(c\sqrt{\log n})$ that no Logspace Turing machine will be able to distinguish from truly random sequences. As a corollary we give an explicit construction of a universal traversal sequence of length $\exp(\exp(c\sqrt{\log n}))$ for arbitrary undirected graphs on n vertices. We then apply the multiparty protocol lower bounds to derive several new time–space trade-offs. We give a tight time–space trade-off of the form $TS = \Theta(n^2)$, for general, k-head Turing machines; the bounds hold for a function that can be computed in linear time and constant space by a $k+1$-head Turing machine. We also give a new length–width trade-off for oblivious branching programs; in particular, our bound implies new lower bounds on the size of arbitrary branching programs, or on the size of Boolean formulas (over an arbitrary finite base). Using universal hashing, Nisan has recently constructed considerably improved random generators for Logspace, with the implication of shorter explicit universal traversal sequences. The time–space and related trade-off results mentioned above are not affected by this development. © 1992 Academic Press, Inc.

* Partly supported by NSF Grant CCR-8706518. Research was done while the third author was a student at the University of Chicago.

† Part of this research was done while the second author was a student at U.C. Berkeley, supported by NSF Grant CCR-8411954.

1. Introduction

This paper is composed of three parts. The first part considers a multiparty communication game and gives some lower bounds on the complexity in this model. The next two parts give applications of these bounds. The second part shows how to construct pseudorandom generators for Logspace and in the third part we obtain some new time–space trade-offs.

1.1. *Multiparty Communication Complexity*

Chandra, Furst, and Lipton [CFL] introduced the following multiparty communication game: Let $f(x_1, ..., x_k)$ be a Boolean function that accepts k arguments each n bits long. k parties wish to collaboratively evaluate f; the ith party knows each input argument except x_i; and each party has unlimited computational power. They share a blackboard, viewed by all parties, where they can exchange messages. The objective is to minimize the number of bits written on the board.

Definition. The *multiparty communication complexity* of f, $C(f)$ is the minimum number of bits that need to be exchanged in the worst case, by any k-party protocol which computes f.

We shall also be interested in the number of bits needed in order to compute f correctly on *most* inputs.

Definition. The *ε-distributional communication complexity* of f, $C_\varepsilon(f)$ is the minimum number of bits that need to be exchanged in the worst case by any k-party protocol that computes f correctly on a $(1+\varepsilon)/2$ fraction of the inputs.

For the special case $k=2$, this multiparty game is exactly the game standard in communication complexity theory, where one party knows x_1, and the other x_2. This case has been extensively investigated in many different contexts and many different lower bounds appear in the literature ([AUY, Y1, BFS], and many more). The distributional communication complexity has been studied as well. Yao [Y3] first considered the distributional communication complexity and proved a lower bound of $\Omega((\log n)^2)$ for the "inner product mod 2" function. Vazirani [Va] improved this bound to $\Omega(n/\log n)$, and Chor and Goldreich [CG] improved it to $\Omega(n)$.

For other values of k less is known. Chandra, Furst, and Lipton [CFL] considered the complexity of the function E_N defined by $E_N(x_1, ..., x_k)=1$ iff $x_1+x_2+\cdots+x_k=N$. For $k=3$ they showed that E_N has a communication complexity of $\Omega(\sqrt{\log N})$. For general k they gave very slowly growing lower bounds, showing that the complexity tends to infinity. For the distributional communication complexity, no previous lower bounds were known.

We prove lower bounds to the distributional communication complexity of two functions. Let

$$f_1(x_1, ..., x_k)=1 \qquad \text{iff } x_1+\cdots+x_k \text{ is a quadratic residue mod } p$$

and

$f_2(x_1, ..., x_k) = 1$ iff the number of positions where all the x_i's have 1 is odd.

We call f_2 the *k-wise inner product* function.

THEOREM 1. *For any n-bit long prime number p*:

$$C_\varepsilon(f_1) = \Omega\left(\frac{n}{2^k} + \log \varepsilon\right).$$

THEOREM 2.

$$C_\varepsilon(f_2) = \Omega\left(\frac{n}{4^k} + \log \varepsilon\right).$$

Theorem 2 is close to best possible. V. Grolmusz [Gr] has recently given a protocol demonstrating that

$$C(f_2) = O\left(k\left\lceil\frac{n}{2^k}\right\rceil\right).$$

The major open problem is to prove lower bounds that do not decrease exponentially in k for some explicit function. Such bounds would give considerable improvements in the applications. Recent results of [Y4, HG] indicate another motivation for such bounds: for any function in ACC^0 (constant depth, polynomial size circuits using "and," "or," and "mod N" gates for some integer N) there exist constants c, d such that the distributional multiparty communication complexity for $k = (\log n)^c$ players is $O((\log n)^d)$. Thus good lower bounds for the distributional (or, in fact, even randomized) multiparty communication complexity of a function f would prove that f is not in ACC^0. For comparison, we should mention that if the Boolean function f is selected at random (uniformly from all functions $\{0, 1\}^{nk} \to \{0, 1\}$) then with high probability,

$$C_\varepsilon(f) \geq \frac{n}{2} - \frac{\log k}{2} + \log \varepsilon.$$

1.2. *Pseudorandom Generators for Logspace*

In the last few years there has been much research directed towards the construction of *pseudorandom* sequences—sequences that "appear" random to some large class of statistical tests. Blum and Micali [BM], Yao [Y1], and, recently, Impagliazzo, Levin, and Luby [ILL] have designed sequences that pass all polynomial time computable tests under some unproven "hardness" assumptions. Reif and Tygar [RT] constructed sequences that pass all NC tests, again under some unproven "hardness" assumption. Ajtai and Wigderson [AW] give sequences that pass all AC^0 tests, using the known "hardness" results for AC^0. Nisan and Wigderson [NW] give general constructions of sequences that pass all tests from a complexity class C, given any function that is "hard" for C. In particular, Nisan constructs pseudorandom sequences for AC^0 using the "hardness" of the parity function [Ni3].

Unfortunately, no lower bounds, "hardness" results, are known for any complexity class but AC^0. Thus in particular no sequences are provably pseudorandom for any class but AC^0.

In this paper we show how to construct sequences that pass all statistical tests that can be performed by Logspace Turing machines (or generally any "small"-space machines). We do this without relying on any unproven "hardness" assumption, but rather by using the multiparty communication complexity lower bounds.

An exact definition of pseudorandom generators for Logspace appears in Section 3. In general, the definition ensures that the output of a pseudorandom generator for Logspace may always be used instead of truly random sequences in any randomized Logspace computation.

THEOREM 3. *There exists an (explicitly given) pseudorandom generator G for Logspace which stretches a seed of* $2^{O(\sqrt{\log n})}$ *random bits to n bits. Moreover, G can be computed in polynomial time (actually, even in Logspace, given multiple access to the input bits).*

A particularly interesting class of tests that can be performed in RL are walks on graphs. The output of our generator will behave just like a random walk on any graph. This allows us to give an explicit construction of universal traversal sequences for general graphs.

THEOREM 4. *There exists an (explicitly given) universal traversal sequence for general undirected graphs of length* $2^{2^{O(\sqrt{\log n})}}$. *Moreover the sequence can be constructed by a Turing machine running in space logarithmic in the length of the sequence.*

Remark. Recently Nisan [Ni2] constructed stronger pseudorandom generators based on universal families of hash functions. These generators convert a random seed of length $S \log R$ into a string of length R that looks random to any SPACE(S)-bounded computation. In particular, out of random seeds of length $O(\log^2 n)$ one obtains strings of polynomial length that look random to Logspace machines.

This result and its consequences, including explicit universal traversal sequences of length $n^{O(\log n)}$, supersede Theorems 3 and 4 and could be matched by the techniques of this paper only if much stronger ($\Omega(n/k)$) lower bounds for the communication complexity of some Logspace computable function were available.

On the other hand, the time–space and related trade-off results to be described in Subsection 1.3 are not affected by the results of [Ni2].

We should mention another recent application of Theorem 2. Luby *et al.* [LVW] construct pseudorandom sequences that fool depth-2 circuits over GF[2] (explicit multivariate polynomials). Like in Theorem 3, their pseudorandom sequence of length n is generated from a random seed of length $2^{O(\sqrt{\log n})}$. But their construction is based on a different principle and mimics the idea of [Ni3, NW]. The requisite hardness result is provided by our lower bound for the k-wise inner product; no substitute is known to work.

1.3. *Time–Space Trade-offs*

1.3.1. *Turing Machines*

The first time–space trade-off we give is for general Turing machines. The machines have a read-only input tape, as well as an arbitrary number of read–write work tapes which count as space. For Turing machines that have only one head on the input tape, quadratic time–space trade-offs were given in 1966 in Cobham's classic paper [Co], and apply, e.g., to such simple languages as the language of all palindromes. However, when the Turing machine is allowed to have two or more heads on the read-only tape, the situation is more complicated.

For multihead Turing machines the known time–space trade-offs are not tight and are significantly sub-quadratic. Duris and Galil [DG] and Karchmer [Ka] exhibit languages that require a time–space trade-off of $T^2S \geqslant \Omega(n^3)$. Gurevich and Shelah [GS] give a language that requires a time–space trade-off of $T^2S \geqslant \Omega(n^{2.5})$ on deterministic machines, but can be solved in linear time and logarithmic space nondeterministically.

We give a very simple proof, that relies on our multiparty communication complexity lower bounds, of a tight, quadratic, time–space trade-off for multihead Turing machines. We actually prove a tight separation between the power of k-head and $(k+1)$-head Turing machines:

THEOREM 5. *The $(k+1)$-wise inner product function over n bit strings requires a time–space trade-off of $TS = \Theta(n^2)$ on any k-head Turing machine.*

Note that this function can be computed in linear time and constant space on a $(k+1)$-head Turing machine. Note also that this lower bound, as well as the separation proved between k-head and $(k+1)$-head Turing machines, is tight as even a 1-head Turing machine can simulate a $(k+1)$-head Turing machine with only a quadratic penalty in time and no penalty in space. Our bounds apply to the nondeterministic, probabilistic, and average complexities as well.

1.3.2. *Branching Programs and Formula Size*

Our next time–space trade-off is for Boolean branching programs. Although the improvements we have here over known results are rather modest (an extra $\log n$ factor), we believe that the techniques are interesting. In particular, significantly stronger bounds would follow if one could improve the lower bounds for multiparty communication complexity.

A branching program is a directed acyclic graph with one source and with two kinds of vertices: vertices that are labeled by an input variable; in this case they must have out-degree two and one of the out-going edges must be labeled with 0 and the other with 1; the other kind of vertices are sinks and they are labeled with

1 or 0. The branching program computes a Boolean function in the natural way. The *size* of the branching program is the number of vertices. A branching program is called *leveled* if the vertices are partitioned into subsets, $L_1, ..., L_l$, called the levels, such that all the edges out of L_i go to L_{i+1}. The number of levels is called the *length* of the branching program, and the size of the largest level is called the *width*. A leveled branching program is called *oblivious* if all the vertices on a single level are labeled by the same variable.

The log of the width of a branching program corresponds to the space, and the length corresponds to the time. In [CFL] it was observed that lower bounds for multiparty communication complexity imply lower bounds on the length of constant width branching programs. Their techniques, however, do not give any nontrivial bounds when the width is allowed to be linear. Currently there is no known result which would given even a $2n$ lower bound on the length of general *polynomial width* branching programs. The best result along these lines is due to Alon and Maass [AM] which gives an $n \log n$ lower bound on the length of polynomial width *oblivious* branching programs.

We generalize the techniques of [AM] to take advantage of multiparty communication complexity lower bounds and achieve stronger bounds. We present an explicit function f^* (in Logspace) such that:

THEOREM 6. *Any oblivious branching program of length $o(n \log^2 n)$ computing f^* requires exponential size.*

This time–space trade-off implies some new lower bounds for general branching program size and for Boolean formula size. So far there was only one lower bound technique known that achieves bounds above $n \log n$, either for the size of formulas over an arbitrary basis, or for general branching program size. This is the nearly quadratic bound $(n^2/\log n)$ due to Nečiporuk [Ne]. We give a new technique for proving such lower bounds. Although the lower bounds we obtain are weaker than Nečiporuk's, they apply to a different class of functions, and the technique we use might be of interest.

COROLLARY 4.4. *Any branching program computing f^* requires $\Omega(n \log^2 n)$ size.*

To obtain lower bounds on formula size, we show that formulas over an arbitrary basis can be converted to branching programs of polynomial size, and of length similar to that of the formula.

COROLLARY 4.5. *Any boolean formula over an arbitrary finite basis that computes f^* requires $\Omega(n \log^2 n)$ size.*

2. MULTIPARTY COMMUNICATION COMPLEXITY

2.1. *Definitions*

Chandra, Furst, and Lipton [CFL] introduced the following multiparty communication game: Let $f(x_1, ..., x_k)$ be a Boolean function that accepts k arguments each n bits long. There are k parties, each having unlimited computational power, who wish to collaboratively evaluate f. The ith party knows all the input arguments *except* x_i. They share a blackboard, viewed by all parties, where they can exchange messages. The objective is to minimize the number of bits written on the board.

The game proceeds in rounds. In each round some party writes one bit on the board. The last bit written on the board is considered to be the outcome of the game and should be the value of $f(x_1, ..., x_k)$. The protocol specifies which party does the writing and what is written in each round. It must specify the following information for each possible sequence of bits that is written on the board so far:

(1) Whether the game is over, and in case it is not over, which party writes the next bit: this should be completely determined by the information written on the board so far.

(2) What that party writes: this should be a function of the information written on the board so far and of the parts of the input that the party knows.

DEFINITION. The *cost* of a protocol is the number of bits written on the board for the worst case input. The *multiparty communication complexity of* f, $C(f)$, is the minimal cost of a protocol that computes f.

We shall also be interested in the number of bits needed in order to compute f correctly on *most* inputs.

DEFINITION. The *bias* a protocol P achieves on f, $B(P, f)$, is defined to be

$$B(P, f) = |\Pr[P(x) = f(x)] - \Pr[P(x) \neq f(x)]|,$$

where $x = (x_1, ..., x_k)$ is chosen uniformly over all k-tuples of n-bit strings.

DEFINITION. The *ε-distributional communication complexity* of f, $C_\varepsilon(f)$ is the minimal cost of a protocol which achieves a bias of at least ε on f.

Let us mention that it is possible to define natural probabilistic and nondeterministic versions of the multiparty communication complexity. As we shall see in Section 2.6, our techniques are strong enough to give the same lower bounds for all these complexity measures (Subsection 2.6).

These measures of complexity refer to worst case inputs. There are natural corresponding concepts of *average case* complexities, and we shall see that similar lower bounds apply to them.

2.2. *Cylinder Intersections*

In this subsection we study the basic structure that a multiparty protocol induces on the set of possible inputs, the set of k-tuples.

Consider a multiparty protocol for evaluating a function. For every possible

k-tuple $x=(x_1, ..., x_k)$ a certain communication takes place, and some string is written on the board. The k-tuples may be partitioned according to the string that gets written on the board.

DEFINITION. Let s be a string and P a multiparty protocol. The s-component, $X_{P,s}$, is defined to be the set of k-tuples $x \in (\{0,1\}^n)^k$ such that on input x the protocol P results in exactly s being written on the board.

The s-components have a special structure, which we will now specify.

DEFINITION. A subset S of k-tuples is called a *cylinder in the* ith *dimension*, if membership in S does not depend on the ith coordinate. A subset of k-tuples is called a *cylinder intersection* if it can be represented as an intersection of cylinders.

LEMMA 2.1. *For any protocol P and string s, $X_{P,s}$ is a cylinder intersection.*

Proof. Define S_i to be the set of k-tuples that is consistent with the communication string from the ith party's point of view. I.e.,

$$S_i = \{(x_1, ..., x_k) : \text{for } \textit{some } x_i', (x_1, ..., x_i', ..., x_k) \in X_{P,s}\}.$$

It is clear that for each i, S_i is a cylinder in the ith coordinate. We shall show that $X_{P,s} = \bigcap_i S_i$.

It is clear that $X_{P,s} \subset \bigcap_i S_i$, so it remains to show that $\bigcap_i S_i \subset X_{P,s}$. Let $(x_1, ..., x_k) \in \bigcap_i S_i$. Then for every i there exists x_i' such that $(x_1, ..., x_i', ..., x_k) \in X_{P,s}$. We claim that on input $(x_1, ..., x_k)$ the protocol will write s on the board. The reason is that all through the communication process the ith party cannot distinguish between the input of $(x_1, ..., x_k)$ and the input of $(x_1, ..., x_i', ..., x_k)$. Thus, the bits written on the board will never deviate from s. ∎

Given a protocol which computes a function f, the value of f must be constant over any single s-component. Our lower bounds will be based upon the fact that for our particular functions f, any cylinder intersection *must* contain approximately the same number of 1's and 0's of the function.

DEFINITION. Let $f\colon (\{0,1\}^n)^k \to \{0,1\}$ be a boolean function. The *discrepancy* of f is

$$\Gamma(f) = \max_S |\Pr[f(x)=1 \text{ and } x \in S] - \Pr[f(x)=0 \text{ and } x \in S]|,$$

where S ranges over all cylinder intersections and x is chosen uniformly over all k-tuples.

LEMMA 2.2. *For any function f,*

$$C(f) \geq \log_2 \left(\frac{1}{\Gamma(f)}\right)$$

and

$$C_\varepsilon(f) \geqslant \log_2\left(\frac{\varepsilon}{\Gamma(f)}\right).$$

Proof. Consider a protocol P achieving a bias of ε on f. We can compute the bias of P on f as the sum of the biases achieved on the different s-components,

$$\begin{aligned}\text{Bias}(P, f) &= |\Pr[P(x) = f(x)] - \Pr[P(x) \neq f(x)]| \\ &\leqslant \sum_s |\Pr[P(x) = f(x) \text{ and } x \in X_{P,s}] - \Pr[P(x) \neq f(x) \text{ and } x \in X_{P,s}]|,\end{aligned}$$

where s ranges over all the possible strings that may be written on the board by the protocol.

For any $x \in X_{P,s}$, $P(x)$ was defined to be the last bit of s, thus

$$\begin{aligned}&|\Pr[P(x) = f(x) \text{ and } x \in X_{P,s}] - \Pr[P(x) \neq f(x) \text{ and } x \in X_{P,s}]| \\ &\quad = |\Pr[f(x) = 1 \text{ and } x \in X_{P,s}] - \Pr[f(x) = 0 \text{ and } x \in X_{P,s}]|.\end{aligned}$$

Since $X_{P,s}$ is a cylinder intersection, we find that the last quantity is bounded from above by $\Gamma(f)$. Thus, if M is the number of different possible strings that may be written on the board by the protocol P, we obtain

$$\text{Bias}(P, f) \leqslant M \cdot \Gamma(f).$$

The statement of the lemma follows, since to produce M different strings requires at least $\log_2 M$ bits. ▮

2.3. *The Discrepancy of Near Hadamard Matrices*

In this subsection we give some motivation to the proof technique for the discrepancy bounds to follow in the subsequent two subsections. Although the direct implications of the observations below pertain to the two-party situation only, it may be instructive to see the root from which the general idea evolved.

Our lower bounds for multiparty protocols will be based on discrepancy estimates over cylinder intersections (Lemma 2.2; see also Subsection 2.6). The situation for two parties is greatly simplified by the fact that a "cylinder intersection" in dimension 2 is simply a rectangle, i.e., a submatrix.

An $N \times N$ matrix A with $(1, -1)$-entries is called an *Hadamard matrix*, if $A^t A = NI$, where I is the identity matrix and A^t is the transpose of A.

For $N = 2^n$, an $N \times N$ Hadamard matrix can be constructed as follows: let the rows and columns be labeled by the n-dimensional vectors over GF(2), the field of two elements; and let the entry in position (x, y) be $(-1)^{x \cdot y}$, where $x \cdot y$ denotes the standard inner product $\sum x_i y_i$ (over GF(2)).

The Chor–Goldreich proof of the $\Omega(n)$ lower bound for the (two-party) distributional communication complexity of the "inner product mod 2" function is based on the observation, attributed to J. H. Lindsey (see [ES, p. 88]), that the sum of the entries in any $r \times s$ rectangle of an $N \times N$ Hadamard matrix is at most $\sqrt{rsN}$. The matrix corresponding to the "inner product modulo 2" function being Hadamard, the result applies directly. Chor and Goldreich also point out that the $p \times p$ matrix (p an n-digit prime) with (i, j) entry $((i-j)/p)$ (Legendre symbol) is "nearly Hadamard" and therefore a similar argument proves an $\Omega(n)$ lower bound on its distributional communication complexity.

The results of the next two subsections will generalize these observations to k parties; and the proof of Lindsey's result for "nearly Hadamard" matrices, essentially a Cauchy–Schwarz argument, is at the root of the idea of the proofs.

We review the generalized Lindsey inequality. We use $\|u\|$ to denote the Euclidean norm of the vector u, and $\|B\|_\infty$ to denote max $|b_{ij}|$, where $B = (b_{ij})$ is a matrix. The entries of the matrix A in the proposition below are arbitrary complex numbers. The asterisk denotes conjugate-transpose; so for a vector u we have $\|u\| = \sqrt{u^*u}$; and if the entries of A are real then $A^* = A^t$.

PROPOSITION 2.3. *Let $A = (a_{ij})$ be an $N \times N$ matrix and $D = \|A^*A - NI\|_\infty$. Let R, $S \subseteq \{1, ..., N\}$, $|R| = r$, $|S| = s$. Let T be the sum of entries in the rectangle $R \times S$, i.e.,*

$$T = \sum_{i \in R} \sum_{j \in S} a_{ij}.$$

Then

$$|T| \leqslant \sqrt{rsN} \cdot \sqrt{1 + Ds/N} \leqslant \sqrt{rsN} \cdot \sqrt{1 + D}.$$

Remarks. In the case of Hadamard matrices $D = 0$, so we obtain the upper bound $\sqrt{rsN}$. "Small perturbations" typically yield near Hadamard matrices with $D = O(\sqrt{N})$; in this case $|T| = O(N^{7/8})$. It is this latter bound that will be generalized in the next two subsections to the discrepancy of the k-dimensional matrices describing the "generalized inner product" and the "quadratic character of the sum of coordinates" functions.

For $(1, -1)$-matrices, the quantity D measures the deviation from Hadamardness. Since the proposition does not restrict the entries of A to ± 1, it is actually more natural to consider its deviation from being unitary. Let $D_0 = \|A^*A - I\|_\infty$ denote this quantity. Now the proposition is equivalent to the following statement:

$$|T| \leqslant \sqrt{rs} \cdot \sqrt{1 + D_0 s}.$$

(To see the equivalence, divide A by $\sqrt{N}$.)

Proof. Let v and w denote the incidence vectors of R and S, respectively, i.e., $(0, 1)$-vectors of length N indicating the elements of R and S. Then $T = v^*Aw$. It follows by the Cauchy–Schwarz inequality that $|T| \leqslant \|v\| \cdot \|Aw\|$. We note that $\|v\| = \sqrt{r}$ and estimate the other term,

$$\|Aw\|^2 = w^*A^*Aw \leqslant Nw^*w + |w^*(A^*A - NI)w| \leqslant Ns + Ds^2.$$

The stated inequality now follows. ▮

We state a corollary regarding two-party distributional complexity. A Boolean function f in $2n$ variables can be described by a $2^n \times 2^n$ $(0, 1)$-matrix. Replacing each zero by -1, we obtain the $(1, -1)$-matrix representing f.

COROLLARY 2.4. *If the $2^n \times 2^n$ $(1, -1)$-matrix representing the Boolean function f is near Hadamard with $D = O(N^{1-c})$ for some constant $c > 0$ $(N = 2^n)$, then $C_\varepsilon(f) = \Omega(n)$.*

2.4. *A Lower Bound for Generalized Inner Product*

In this subsection we prove a lower bound on the multiparty communication complexity of the generalized inner product (GIP). The bound will follow via Lemma 2.2 from an upper bound on the discrepancy of GIP over cylinder intersections (Lemma 2.5).

DEFINITION. The *k-wise generalized inner product* function of n-bit strings $x_1, ..., x_k$ is defined by

$\mathrm{GIP}_{n,k}(x_1, ..., x_k) = 1$ iff the number of positions in which each x_i has 1 is odd.

We first introduce the ± 1-version of GIP for easier handling of the discrepancy.

DEFINITION. $f(x_1, ..., x_k)$ is 1 if $\mathrm{GIP}(x_1, ..., x_k) = 0$ and -1 if $\mathrm{GIP}(x_1, ..., x_k) = 1$.

DEFINITION. Let

$$\Delta^k(n) = \max_{\phi_1, ..., \phi_k} \Big| \mathop{E}_{x_1, ..., x_k} f(x_1, ..., x_k)\, \phi_1(x_1, ..., x_k) \cdots \phi_k(x_1, ..., x_k) \Big|,$$

where the maximum is taken over all functions ϕ_i: $(\{0, 1\}^n)^k \to \{0, 1\}$ s.t. ϕ_i does not depend on x_i.

The E stands for expected value over all the possible 2^{nk} choices of $x_1, ..., x_k$. Note that $\Delta^k(n)$ is exactly $\Gamma(\mathrm{GIP}_{k,n})$, the discrepancy of the k-wise inner product function on n-bits.

LEMMA 2.5.

$$\Gamma(\mathrm{GIP}_{k,n}) = \Delta^k(n) \leqslant \mu_k^n,$$

where μ_k is given by the recursion : $\mu_1 = 0$, and $\mu_k = \sqrt{(1 + \mu_{k-1})/2}$.

Note. It can be shown by induction that $\mu_k \leqslant 1-4^{1-k} < e^{-4^{1-k}}$.

Proof. We proceed by induction on k. It is clear that $\Delta^1(n)=0$, except for the case $n=0$, where we obtain 1 (let us define $0^0=1$ for this paper).

Let $k \geqslant 2$. Since ϕ_k does not depend on x_k and $|\phi_k| \leqslant 1$,

$$\Delta^k(n) \leqslant \mathop{E}_{x_1, ..., x_{k-1}} |\mathop{E}_{x_k} f(x_1, ..., x_k)\, \phi_1(x_1, ..., x_k) \cdots \phi_{k-1}(x_1, ..., x_k)|.$$

In order to estimate the right-hand side, we shall use the following version of the Cauchy–Schwarz inequality:

Cauchy–Schwarz Inequality. For any random variable z: $E[z]^2 \leqslant E[z^2]$.

Thus our estimate is

$$\Delta^k(n) \leqslant \mathop{E}_{x_1, ..., x_{k-1}} [\mathop{E}_{x_k} f(x_1, ..., x_k)\, \phi_1(x_1, ..., x_k) \cdots \phi_{k-1}(x_1, ..., x_k]^2]^{1/2}$$
$$= [\mathop{E}_{u, v, x_1, ..., x_{k-1}} f(x_1, ..., x_{k-1}, u)\, f(x_1, ..., x_{k-1}, v)\, \phi_1^u \phi_1^v \cdots \phi_{k-1}^u \phi_{k-1}^v]^{1/2},$$

where ϕ_i^u stands for $\phi_i(x_1, ..., x_{k-1}, u)$, and ϕ_i^v for $\phi_i(x_1, ..., x_{k-1}, v)$.

To estimate this we shall need the following observation: For every particular choice of u and v, $f(x_1, ..., x_{k-1}, u)\, f(x_1, ..., x_{k-1}, v)$ can be expressed in terms of the function f on $k-1$ strings of a possibly shorter length. Inspection reveals that $f(x_1, ..., x_{k-1}, u)\, f(x_1, ..., x_{k-1}, v)$ is simply $f(z_1, ..., z_{k-1})$, where z_i is the restriction of x_i to the coordinates j such that $u_j \neq v_j$. We shall now view each x_i as composed of two parts: z_i and y_i, where z_i is the part of the x where $u_j \neq v_j$, and y_i is the rest (this is done separately for every u, v).

For every particular choice of u, v and consequently $y_1, ..., y_{k-1}$, we define a function of the "z-parts",

$$\xi_i^{u, v, y_1, ..., y_{k-1}}(z_1, ..., z_{k-1}) = \phi_i(x_1, ..., x_{k-1}, u)\, \phi_i(x_1, ..., x_{k-1}, v),$$

where each x_i is obtained by the concatenation of the corresponding y_i and z_i. We can now rewrite the previous estimate as

$$\Delta^k(n) \leqslant [\mathop{E}_{u, v} \mathop{E}_{y_1, ..., y_{k-1}} S^{u, v, y_1, ..., y_{k-1}}]^{1/2},$$

where $S^{u, v, y_1, ..., y_{k-1}}$ is defined by

$$S^{u, v, y_1, ..., y_{k-1}} = \mathop{E}_{z_1, ..., z_{k-1}} f(z_1, ..., z_{k-1})\, \xi_1^{u, v, y_1, ..., y_{k-1}}(z_1, ..., z_{k-1}) \cdots$$
$$\cdots \xi_{k-1}^{u, v, y_1, ..., y_{k-1}}(z_1, ..., z_{k-1}).$$

Now, $S^{u, v, y_1, ..., y_{k-1}}$ can be estimated via the induction hypothesis. Indeed note that $\xi_i^{u, v, y_1, ..., y_{k-1}}$ does not depend on z_i. Thus the previous estimate of $\Delta^k(n)$ is bounded by

$$\Delta^k(n) \leqslant [\underset{u, v, y_1, ..., y_{k-1}}{E} \Delta^{k-1}(m_{u,v})]^{1/2}$$
$$\leqslant [\underset{u, v, y_1, ..., y_{k-1}}{E} \mu_{k-1}^{m_{u,v}}]^{1/2},$$

where $m_{u,v}$ is the length of the strings z_i, which is equal to the number of positions j such that $u_j \neq v_j$.

Since u and v are distributed uniformly in $\{0, 1\}^n$, $m_{u,v}$ is distributed according to the binomial distribution. For any constant m, the probability that $m_{u,v} = m$ is exactly $\binom{n}{m} 2^{-n}$. Thus the previous estimate is given by:

$$\Delta^k(n) \leqslant \left[\sum_{m=0}^{n} \binom{n}{m} 2^{-n} \mu_{k-1}^m\right]^{1/2}$$
$$= [2^{-n}(1+\mu_{k-1})^n]^{1/2} = \mu_k^n.$$

This completes the proof of the lemma. ∎

Combining this bound with Lemma 2.2 we obtain:

THEOREM 2.

$$C_\varepsilon(\mathrm{GIP}_{k,n}) = \Omega\left(\frac{n}{4^k} + \log_2 \varepsilon\right).$$

2.5. *A Lower Bound for the Quadratic Character*

In this section we prove a lower bound on the multiparty communication complexity of the "quadratic character of the sum of coordinates" function (QCS).

DEFINITION. Let $\mathrm{QCS}_{p,k}(x_1, ..., x_k)$ be $((x_1 + \cdots + x_k)/p)$, where p is an n-bit prime, $x_1, ..., x_k$ are n-bit integers (zeros allowed on the left), and (x/p) denotes the Legendre symbol.

As in the previous subsection, the lower bound on the communication complexity of $\mathrm{QCS}_{p,k}$ will follow via Lemma 2.2 from an upper bound on the discrepancy of this function over cylinder intersections. The discrepancy bound in turn will be derived by extending character sum estimates of A. Weil.

Let χ be a multiplicative character of order $d > 1$ over F_p. (In our applications $d = 2$ and $\chi(x) = (x/p)$ is the Legendre symbol.) Let $g(x)$ be a polynomial over F_p. We say that g is a constant times a full dth power if $g(x) = c \cdot (h(x))^d$ for some $c \in F_p$ and $h(x) \in F_p[x]$. We shall use the following estimate from the theory of character sums (cf. Schmidt [Sch, p. 43, Theorem 2C]).

THEOREM (A. Weil). *If $g \in F_p[x]$ has m distinct roots in the algebraic closure of F_p and g is not a constant times a full dth power then*

$$\sum_{x \in F_p} \chi(g(x)) \leqslant (m-1)\sqrt{p}. \tag{1}$$

Our chief tool will be a generalization of this result. We replace x by $\sum_{i=1}^{k} x_i$ and extend the summation over a subdomain of F_p^k only. The subdomain will be an intersection of k cylinders.

LEMMA 2.6. *Let p be an odd prime and d, s, k be positive integers. Assume $d \geqslant 2$ and $s < p$. Let $\chi: F_p \to \mathbf{C}$ be a multiplicative character of degree d and let $g \in F_p[x]$. If g has at most s distinct roots in the algebraic closure of F_p and g is not constant times a full dth power, then for arbitrary $(0, 1)$-valued functions $\phi_1(x_1, ..., x_k)$, ..., $\phi_k(x_1, ..., x_k)$, $\phi_i: F_p^k \to \{0, 1\}$, such that ϕ_i does not depend on the ith variable, we have*

$$\left| \sum_{x_1 \in F_p} \sum_{x_2 \in F_p} \cdots \sum_{x_k \in F_p} \chi(g(x_1 + \cdots + x_k))\, \phi_1 \cdots \phi_k \right| \leqslant c(s,k) \cdot p^{k - 2^{-k}}, \tag{2}$$

where $c(s,k) = 2 \cdot ((2^{k-1}s - 1)/2)^{2^{-(k-1)}}$.

Remark. Observe that for $s < k$, we have $c(s,k) < 3$. In our applications, $s = 1$. We need the more general result for the sake of the proof by induction.

Proof. We first observe that g must satisfy the following condition:

Condition (*). For some root α of g in the algebraic closure of F_p, not all elements $\alpha + t$, $t \in F_p$, have the same multiplicity modulo d.

Indeed, this follows from the two assumptions that $s < p$ and g is not a constant times a full dth power. For the inductive proof, we shall drop these two assumptions and assume Condition (*) only. (This, of course, still implies that g is not a constant times a full dth power, but weakens the $s < p$ condition.)

Now we proceed by induction on k. The case $k = 1$ is precisely Weil's character sum estimate (1).

Assume $k \geqslant 2$. Let S denote the sum to be estimated. Then, clearly

$$|S| \leqslant \sum_{x_1, ..., x_{k-1} \in F_p} \left| \sum_{x_k \in F_p} \chi(g(x_1 + \cdots + x_k))\, \phi_1 \cdots \phi_{k-1} \right|.$$

We use the Cauchy–Schwarz inequality to estimate the right-hand side:

$$|S| \leqslant p^{(k-1)/2} \cdot \left(\sum_{x_1, ..., x_{k-1} \in F_p} \left(\left| \sum_{x_k \in F_p} \chi(g(x_1 + \cdots + x_k))\, \phi_1 \cdots \phi_{k-1} \right| \right)^2 \right)^{1/2}. \tag{3}$$

We expand the squares on the right-hand side. The following notation will be convenient:

$$F^{u,v}(x) = g(x+u)\, g(x+v);$$
$$\psi_i^{u,v}(x_1, \ldots, x_{k-1}) = \phi_i(x_1, \ldots, x_{k-1}, u)\, \phi_i(x_1, \ldots, x_{k-1}, v).$$

Observe that $\psi_i^{u,v}$ does not depend on x_i. With this notation the right-hand side of inequality (3) expands to

$$p^{(k-1)/2} \cdot \sum_{u, v \in F_p} S(u, v), \tag{4}$$

where

$$S(u, v) = \sum_{x_1, \ldots, x_{k-1} \in F_p} \chi(F^{u,v}(x_1 + \cdots + x_{k-1}))\, \psi_1^{u,v} \cdots \psi_{k-1}^{u,v}. \tag{5}$$

(We have used u and v to denote the two occurrences of x_k and moved the summation over $(x_1, \ldots, x_{k-1})$ inside.)

For $u \neq v$ we observe that $S(u, v)$ is just an instance of the sum S in (2), with $k-1$ in the place of k and $2s$ in the place of s.

In order to justify this claim, we have to see that $F^{u,v}$ satisfies Condition (*).

Let R be the multiset of roots of g in the algebraic closure of F_p. Then R has a member α defined in Condition (*).

The set of roots of $F^{u,v}$ is the multiset $(R-u) \cup (R-v)$. Let μ_i denote the multiplicity of $\alpha + i(u-v)$ in R. Observe that the multiplicity of $\alpha - u + i(u-v)$ in the multiset $(R-u) \cup (R-v)$ is $\mu_{i-1} + \mu_i$. Therefore if $F^{u,v}$ does not satisfy Condition (*) then for some λ, we have $\mu_{i-1} + \mu_i \equiv \lambda \bmod d$ for every integer i (subscripts are taken $\bmod\, p$). Hence $\mu_{i-1} \equiv \mu_{i+1} \bmod d$. Since p is odd, it follows that all the μ_i are congruent $\bmod\, d$, contrary to the choice of α. This observation sets up the conditions for an inductive step.

We infer by induction that for $u \neq v$

$$|S(u, v)| \leqslant c(2s, k-1)\, p^{k-1-2^{-(k-1)}}.$$

For the case $u = v$ we use the trivial estimate

$$|S(u, u)| \leqslant p^{k-1}.$$

Substituting into Eq. (4) we obtain

$$\begin{aligned} |S| &\leqslant p^{(k-1)/2}(p \cdot p^{k-1} + p(p-1)\, c(2s, k-1)\, p^{k-1-2^{-(k-1)}})^{1/2} \\ &\leqslant \sqrt{2}c(2s, k-1)^{1/2}\, p^{k-2^{-k}} = c(s, k)\, p^{k-2^{-k}}. \end{aligned}$$

This completes the proof of the lemma. ∎

COROLLARY 2.7. *For the discrepancy of the "quadratic character of the sum of coordinates" function we have*

$$\Gamma(\mathrm{QCS}_{p,k}) < 2p^{-2^{-k}}.$$

Proof. We just have to substitute the right parameters in Lemma 2.6. Let $d = 2$, $s = 1$, and let $\chi(x) = (x/p)$. Let the identity function x play the role of g in Lemma 2.6. Now $c(s, k) = c(1, k) < 2$; the order of χ is $d = 2$; the polynomial x is not a full square and has $s = 1$ root. Thus all the conditions are met. The conclusion is that the sum of f over any cylinder intersection is less than

$$2p^{k-2^{-k}}.$$

Dividing by p^k, the size of the space, we obtain the desired bound. ∎

Combining this bound with Lemma 2.2, we obtain:

THEOREM 1. *If p is an n-bit prime, then*

$$C_\varepsilon(\mathrm{QCS}_{p,k}) = \Omega(n2^{-k} + \log \varepsilon).$$

2.6. *Probabilistic and Nondeterministic Protocols and Average Case Complexity*

Upper bounds for the discrepancy over cylinder intersections yield lower bounds for the communication complexity in models far more powerful than the deterministic protocols considered so far, and they extend to *average case complexity*.

Next we consider nondeterministic and probabilistic protocols. In a nondeterministic protocol each player may act nondeterministically. In this model, at each step the protocol may specify a set of possible moves for each player instead of just one possible move. A nondeterministic protocol computes a function f if for any input x such that $f(x) = 1$ there exists a nondeterministic choice for which the protocol outputs 1, and for each x such that $f(x) = 0$ for all nondeterministic choices the protocol outputs 0. The cost of a protocol is the worst case number of bits written over all inputs and all nondeterministic choices, and the nondeterministic complexity of f, $C^N(f)$, is the minimal cost of a nondeterministic protocol that computes f.

The discrepancy may also be used to give lower bounds on the nondeterministic complexity.

LEMMA 2.7. *Let p be the fraction of k-tuples x for which $f(x) = 1$. Then*

$$C^N(f) \geqslant \log_2(p/\Gamma(f))$$

Proof. As in the case of deterministic protocols, for a nondeterministic protocol P and each string s that may be written on the board consider the s-component, $X_{P,s}$, which is the set of inputs x for which some nondeterministic choices result in s being written on the board. As opposed to the deterministic case now the various s-components do not necessarily partition the space of k-tuples. It is still true, however, that each s-component is a cylinder intersection (the proof of Lemma 2.1 carries over). Finally, all the 1's of f must be covered by some component which outputs 1. No such component may contain any zeros of f and thus its size is bounded by $\Gamma(f)$. ∎

A probabilistic protocol is a probability distribution over deterministic protocols. It computes a function f if for every input it produces the correct answer with probability $\frac{2}{3}$ (probability taken over the distribution of deterministic protocols). The cost of a randomized protocol is the worst case over all inputs of the expected number of bits written on the board. The randomized complexity of f, $C^R(f)$, is the minimal cost of a randomized algorithm that computes f. (Note. We consider here a strong notion of randomized protocols; they are allowed to make a bounded two-sided error, and the parties have shared common random coins—the description of the deterministic protocol that was chosen randomly.) The distributional complexity gives a lower bound to the randomized complexity.

LEMMA 2.8.

$$C^R(f) \geqslant C_{1/3}(f) \geqslant \log_2(1/(3\Gamma(f))).$$

Proof. A straightforward averaging argument shows that one of the deterministic protocols making up a probabilistic protocol for f must be correct on at least $\frac{2}{3}$ of the inputs. The rightmost inequality is a restatement of Lemma 2.2. ∎

The deterministic, nondeterministic, and probabilistic models considered so far are all *worst case* models. We may be interested in the median cost (over all inputs) of a (deterministic, nondeterministic, or probabilistic) protocol. One can in fact show that the discrepancy upper bound implies that in each of these models, *almost every input is hard.*

Let X be one of the letters D, N, P, referring to deterministic, nondeterministic, and probabilistic protocols, respectively. The *δ-threshold* of a set S of real numbers is the smallest α such that a δ fraction of the members of S is $\leqslant \alpha$. The *δ-threshold complexity* of a protocol of type X is the δ-threshold of the costs of the protocol over all inputs (all "yes" inputs in the nondeterministic case). The *δ-threshold complexity of a function f in model X* is the minimum of the δ-threshold complexities of all protocols of type X, computing the function f. We denote this quantity by $C^{X,\delta}(f)$. If this quantity is large then a $1-\delta$ fraction of the inputs is hard for protocols of type X.

LEMMA 2.9.

$$C^{D,\delta}(f) \geqslant \log_2(\delta/\Gamma(f)).$$

$$C^{N,\delta}(f) \geqslant \log_2(\delta p/\Gamma(f)).$$

$$C^{R,\delta}(f) \geqslant \frac{c}{|\log \delta|} \cdot \log\left(\frac{\delta}{3\Gamma(f)}\right).$$

(Here, c is an absolute constant <20.)

Proof. The number of homogeneous (all zeros or all ones) cylinder intersections required to cover a δ fraction of the input is at least $\delta/\Gamma(f)$, hence the first inequality.

The argument for the nondeterministic case is the same except we cover the "yes" inputs only.

For the probabilistic case, we repeat the protocol $c\,|\log \delta|$ times and take the majority vote. This ensures that the probability of error will be $< \delta/3$ on every input. Now, as before, by the usual averaging argument we see that one of the deterministic protocols making up the probabilistic protocol must be correct on a $1-\delta/3$ fraction of the inputs. Restricting our attention to the least expensive δ fraction, we see that on that set the protocol still achieves a bias $>\frac{1}{3}$ and therefore has cost $>\log_2(1/3\Gamma(f))$. ▮

We thus obtain that the lower bounds given in Theorems 1 and 2 apply also to randomized and to nondeterministic protocols, even in the average case. In the results below, δ is an arbitrary positive constant.

THEOREM 1′. *If p is an n-digit prime then*

$$C^{N,\delta}(\mathrm{QCS}_{p,k}) = \Omega(n/2^k)$$
$$C^{R,\delta}(\mathrm{QCS}_{p,k}) = \Omega(n/2^k).$$

THEOREM 2′.

$$C^{N,\delta}(\mathrm{GIP}_{k,n}) = \Omega(n/4^k)$$
$$C^{R,\delta}(\mathrm{GIP}_{k,n}) = \Omega(n/4^k).$$

3. Pseudorandom Generators for Logspace

In this section we show how to use our lower bounds for multiparty protocols in order to construct pseudorandom generators for Logspace (or generally any small-space machines).

3.1. *Randomized Space Bounded Computation*

We first review the definition of randomized space bounded computation. There are several subtle points to consider when defining randomized space-bounded classes. We shall present here the "correct" definition. For an overview of the subtleties involved we refer, e.g., to [BCDRT].

DEFINITION. A *randomized, space* $s(n)$ Turing machine is defined to have the following properties:

(1) The Turing machine runs in space $s(n)$ on any input of size n.

(2) The Turing machine may flip a fair unbiased coin at any stage.

(3) The Turing machine may *never* get into an infinite loop, for any sequence of coin flips. In particular, with probability 1 it terminates in time $\exp(s(n))$ on any input of length n.

A Turing machine accepts a language L with one-sided error if for every $x \in L$ the machine accepts with probability at least $\frac{1}{2}$ and for any x not in L it rejects with probability 1. A Turing machine accepts a language L with two-sided error if for any $x \in L$, the Turing machine accepts with probability of at least $\frac{2}{3}$, and for any x not in L it rejects with probability of at least $\frac{2}{3}$.

RSPACE$(s(n))$ is the class of languages accepted with one-sided error by space $s(n)$ randomized Turing machines. BPSPACE $(s(n))$ is the class of languages accepted with two-sided error by randomized space $s(n)$ Turing machines. RL is RSPACE$(O(\log n))$, and BPL is BPSPACE$(O(\log n))$. (Our notation differs slightly from [BCDRT]. They use RLP and BPLP, resp., to denote these classes.)

We want to focus attention on condition (2), the kind of access the machine has to the random bits. As defined, the randomized Turing machine has access to the random bits one by one. When it wants a random bit it can flip a coin, but in no case can it go "back" and review the result of a previous coin flip. Any bit that it wishes to "remember" must be kept in the limited storage. This restriction that the machine does not have multiple, two-way, access to the random bits is essential, as, for example, random-Logspace machines with two-way access to the random bits are not even known to be in deterministic polynomial time (in contrast to RL and BPL being in P).

It is interesting to note that the same apparent difference in power between one-way and two-way access holds also for nondeterministic computation. In the "correct" definition of NL, the Turing machine has one-way access to the nondeterministic bits. If the definition is changed to allow two-way access to nondeterministic bits, then the machines turn out to have the full power of NP.

3.2. *Space Bounded Statistical Tests*

We are interested in producing pseudorandom sequences that might be used instead of truly random sequences in space bounded computations. Thus the statistical tests that must be passed by the generator are the ones that can be performed by a space bounded Turing machine on its *random source.* Note that this class of tests is a proper subset of the tests that can be performed by space bounded machines on their *input tape.*

We shall allow non-uniform statistical tests as well.

DEFINITION. A *space-$s(n)$ statistical test* is a deterministic space $s(n)$ Turing machine M, and an infinite sequence of binary strings $a = (a_1, ..., a_n, ...)$ called the advice strings. We require that the length of a_n is at most $\exp(s(n))$.

The result of the test on input x, $M^a(x)$, is determined as follows: The string a_n, where n is the length of x, is put on a special read-only tape of M called the advice tape. The machine M is run on the advice tape, treating it as a normal input tape. The machine has the following one-way mechanism to access x: at any point it may request the next bit of x (in the same fashion as a randomized Turing machine may request the next random bit).

Note that these tests have considerable power. We next give some examples of operations that can be performed and combined by Logspace tests:

Consider the input as partitioned into consecutive words, each of some fixed small length ($O(\log n)$). The following questions can all be answered by a Logspace test.

- Count the number of times a certain value appears.
- Compute the average value of a word.
- Compute the standard deviation, or higher moments.
- Compute any of the previous measures for an arbitrary subset of the words, or of the bits.

In fact most of the statistical tests described by Knuth [Kn] lie in this class.

3.3. *Pseudorandom Generators for Space Bounded Computation*

A pseudorandom generator for space $s(n)$ must produce strings that look random to any space $s(n)$ statistical test.

DEFINITION. $G = \{G_n : \{0, 1\}^{l(n)} \to \{0, 1\}^n\}$ is called a *pseudorandom generator* for $s(n)$, if for every polynomial $p(n)$, all large enough n, and every space $s(n)$ statistical test M^a,

$$|\Pr[M^a(y) \text{ accepts}] - \Pr[M^a(G(x)) \text{ accepts}]| \leq 1/p(n),$$

where y is chosen uniformly in $\{0, 1\}^n$ and x is chosen uniformly in $\{0, 1\}^{l(n)}$.

The first observation we should make when trying to construct a pseudorandom generator for space-bounded Turing machines is that, without loss of generality, we can restrict the class of statistical tests that have to be passed. In a fashion similar to Yao's [Y2] results for general (polytime-hard) pseudorandom generators, one can show that any generator that passes all space $s(n)$ *prediction* tests will be a pseudorandom generator for space $s(n)$.

DEFINITION. The family $G = \{G_n: \{0, 1\}^{l(n)} \to \{0, 1\}^n\}$ of functions *passes all space-$s(n)$ prediction tests* to within $\varepsilon(n)$ if for every space $s(n)$ statistical test M^a, and every $1 \leq i \leq n$,

$$|\Pr[M^a(\text{first}(i-1) \text{ bits of } G(x)) = i\text{th bit of } G(x)] - \tfrac{1}{2}| \leq \varepsilon(n),$$

where x is chosen uniformly from $\{0, 1\}^{l(n)}$. G is said to pass all space $s(n)$ prediction tests if for any polynomial $p(n)$, G passes all space $s(n)$ prediction tests to within $1/p(n)$ for all sufficiently large n.

LEMMA 3.1. *G is a pseudorandom generator for space $s(n)$ iff it passes all space $s(n)$ prediction tests.*

Proof. Similar to Yao's proof [Y2]. ▌

3.4. *The Generator*

Our generator is based on a function f that takes k arguments, each r bits long, and has high multiparty communication complexity.

THE GENERATOR.

Input. The input to the generator will consist of t boolean strings, each r bits long (all together, rt random bits).

Output. Each output bit will be of the form $f(S)$, where S is some cardinality k subset of the input strings. The order is extremely important, and we shall take *all* the k-subsets in the *antilexicographic* order. (I.e., $f(S_1)$ appears before $f(S_2)$ if the last string in the symmetric difference of S_1 and S_2 belongs to S_2.) All together there are $\binom{t}{k}$ output bits.

LEMMA 3.2. *For any $\varepsilon > 0$ and $s < C_\varepsilon(f)/k$, the above generator passes all space s prediction tests to within ε.*

Proof. Assume not; we shall give a multiparty protocol that predicts f with bias ε using less than $C_\varepsilon(f)$ bits of communication. Suppose the lth bit, $f(S)$, can be predicted by the Turing machine, where $S = \{x_{i_1}, x_{i_2}, ..., x_{i_k}\}$, and $i_1 > i_2 > \cdots > i_k$. We shall now show how k parties, the jth knowing the values of each x_i in S *except* x_{i_j}, can predict $f(x_{i_1}, x_{i_2}, ..., x_{i_k})$, with low communication.

By an averaging argument, it is possible to fix the values of all x_i's not in S to constants in some way, while preserving the prediction bias of the Turing machine. Thus we can assume w.l.o.g. that all x_i's not in S are fixed and known to all the parties beforehand.

The parties will simulate the Turing machine running on the first $l - 1$ bits of the output of the generator. Since the parties have unlimited power, they have no problem simulating the Turing machine as long as they have access to the input bits which the Turing machine reads (or can compute them from the information they have). In our case, however, no single party can do so, since each party is missing the value of x_{i_j} for some j, which is needed to compute some of the input bits (i.e., bits of the form $f(T)$, where T contains x_{i_j}).

In our simulation each party simulates the Turing machine for as long as it can and then sends the current state of the machine to the next party to continue the simulation. The simulation starts with the first party (i.e., the party missing only x_{i_1}). Note that this player can simulate the Turing machine until the first time that a bit involving x_{i_1} is read by the Turing machine. At this point the first player sends the total state of the Turing machine to the second player who continues with the simulation. This player can now continue the simulation until the first point where a bit involving x_{i_2} is read, at which point he sends the state of the machine to the third player, etc. This continues all way to the kth player.

The important thing to notice is that once the last player starts with the simulation, he can continue it until the prediction of $f(S)$. This is so because of the ordering of the bits of the generator that we chose. Consider the point where the first party cannot continue the simulation since a bit involving x_{i_1} first appears. Because the sets used to compute the bits of the generator were ordered in anti-lexicographic order, we observe that all sets from this point on, until S, contain x_{i_1}. Similarly, once x_{i_2} appears it remains in every set until S, etc. So, when the kth party cannot continue the simulation, the next set must contain all of $x_{i_1}, ..., x_{i_k}$; i.e., it must be S that is the point where the Turing machine makes the prediction.

Sending the total state of the Turing machine requires s bits, and since this is done in our simulation only $k-1$ times, we obtain a protocol that requires only $(k-1)s$ bits and predicts $f(S)$ with a bias of ε. This contradicts the choice of $s < C_\varepsilon(f)/k$. ∎

We can now state the main result of this section:

THEOREM 3. *For some constants $c_1, c_2 > 0$, there exists an explicit pseudorandom generator, $G = \{G_n : \{0, 1\}^{l(n)} \to \{0, 1\}^n\}$, for space $2^{c_1\sqrt{\log n}}$, where $l(n) = 2^{c_2\sqrt{\log n}}$. Moreover, G can be computed by a Logspace Turing machine (having multiple access to its input bits).*

Proof. We use the construction of the generator described in this section, with f being the "generalized inner product" function, $k = 2\sqrt{\log n}$, $t = 2^k$, and $r = 8^k$. Theorem 1 guarantees the high communication complexity of this function, and thus Lemma 3.2 shows that the generator passes all prediction tests, and Lemma 3.1 concludes that it is a pseudorandom generator. ∎

3.5. *One-Way vs. Two-way Access to Randomness*

Our generator sheds some light on the difference between one-way and two-way access to the random bits given to Logspace machines. We show that two-way access is better, at least in the sense that fewer random bits are necessary.

COROLLARY 3.3. *A randomized Logspace Turing machine with one-way access to the random bits that uses a polynomial number of random bits can be simulated by a randomized Logspace machine with two-way access to the random bits that uses only $2^{O(\sqrt{\log n})}$ random bits.*

Proof. Our generator can be implemented by a Logspace machine having two-way access to the random bits. The generator can be easily run "on the fly" and can generate one bit at a time to supply to the original, simulated machine, whenever it flips a coin. ▌

Recently, Nisan [Ni1] exhibited another way in which two-way access is better than one-way access; any randomized Logspace machine with one-way access to the random bits which accepts a language with two-sided error can be simulated by a *zero-error* randomized Logspace machine that has two-way access to the random bits.

3.6. *Universal Sequences*

One of the most interesting subclasses of Logspace statistical tests are those related to walks on graphs. Given a graph H on the advice tape, a Logspace machine can treat the input to the test as directions for a walk on the graph and perform the walk. Thus the output of a pseudorandom generator for Logspace will behave like a random walk on any graph. We use this fact to construct universal traversal sequences.

DEFINITION. A graph is called *(d, n)-labeled* if it is a d-regular graph on n vertices and the edges adjacent to each vertex are labeled by a permutation of $\{1, ..., d\}$ (an edge may be labeled differently at each of its two vertices). A string $w \in \{1, ..., d\}^*$ is said to *cover* a (d, n)-labeled graph, if w, when treated as directions to a walk on the graph, visits all vertices on the graph, whatever the starting vertex is.

DEFINITION. A string $w \in \{1, ..., d\}^*$ is said to be a (d, n) *universal traversal sequence* if it covers every (d, n)-labeled graph.

[AKLLR] showed that a random string of length $O(d^2n^3 \log n)$ is a (d, n)-universal sequence with high probability. Results shown in [KLNS] imply that a random string of length $O(dn^3 \log n)$ actually suffices. However, explicit construction of short universal sequences is more difficult. Explicit constructions are known for two special cases: for $d = 2$ an explicit construction is known of polynomial length universal sequences [Is]; and for $d = n - 1$ an explicit construction of length $n^{\log n}$ is known [KPS]. Our pseudorandom sequences allow us to give (d, n)-universal sequences of length $2^{2^{O(\sqrt{\log n})}}$ for *all* values of d.

We shall be using the output of the generator as a random walk. A technical issue that should be mentioned is that we need to convert the *binary* string which is the output of the generator to a string in $\{1, ..., d\}$. One way to do this is to take every $5 \log n$ consecutive bits modulo d. This way a uniform distribution on a binary string will be converted to an almost uniform distribution on walks.

LEMMA 3.4. *Let $G = \{G_n : \{0, 1\}^{l(n)} \to \{0, 1\}^{n^4}\}$ be a pseudorandom generator for Logspace. Then for every (d, n)-labeled graph H, $G(x)$ (converted as mentioned) will cover H with probability at least $\frac{1}{2}$ (probability taken over a random choice of x).*

Proof. The description of H can be put in the oracle, and then a Logspace machine can perform the walk on H given by its input. A truly random string y of length n^4 converted in this manner will result in a nearly uniformly random walk on H and, as such, will visit every vertex, starting from every vertex with probability of at least $1 - 1/3n^2$. A Logspace machine can determine whether vertex i is reached in a walk starting from vertex j, and thus for every i, j, the probability that $G(x)$ (converted to a walk) will reach vertex i starting from vertex j, should be at least $1 - 1/2n^2$. Thus the probability that there exist i, j such that the walk from i does not reach j is at most $\frac{1}{2}$, ∎

LEMMA 3.5. *Let $G = \{G_n : \{0, 1\}^{l(n)} \to \{0, 1\}^{n^4}\}$ be a pseudorandom generator for Logspace. Then the string achieved by the concatenation of $G(x)$ for all possible $2^{l(n)}$ values of x (and converted as mentioned) is a (d, n) universal traversal sequence.*

Proof. For each (d, n)-labeled graph H, half the substrings of the form $G(x)$ will cover H. Thus when one of these substrings is reached, whatever vertex the walk is in, H will be covered by it. ∎

Applying our generator to this lemma we obtain:

THEOREM 4. *For every d and n, there exists an (explicitly given) (d, n) universal traversal sequence of length $2^{2^{O(\sqrt{\log n})}}$. Moreover, the sequence can be constructed by a Turing machine running in space logarithmic in the length of the sequence.*

4. TIME–SPACE TRADE-OFFS

4.1. *Trade-offs for Turing Machines*

LEMMA 4.1. *Let $f(x_1, ..., x_k)$ be a boolean function with multiparty communication complexity of $C(f)$. Then any $(k-1)$-head Turing machine that computes f from the following input:*

$$\langle x_1 \rangle \#\#\#\#\#\#\#\# \langle x_2 \rangle \#\#\#\#\#\#\#\# \cdots \#\#\#\#\#\#\#\# \langle x_k \rangle$$

(where $\#\#\#\#\#\#\#$ means l spaces on the input tape) requires a time–space trade-off of $TS \geq l \cdot C(f)/k$.

Proof. We shall simulate the machine by a k-party protocol. At any point in time the $k-1$ heads may lie in at most $k-1$ different x_i's (some heads may lie in the same x_i or in the space between adjacent x_i's). At that point in time the TM can be simulated by the party who has these $k-1$ x_i's (provided that party has the total state of the TM). This party can continue to simulate the TM until some head moves into the region of the "missing" x_i. We call this event a *transition*. In this case the simulation can be continued by a different party, one that now has all the current x_i's. Thus the current simulating party will send the total state of the TM to the new simulator, who will continue the simulation until the next transition.

The number of bits communicated at each transition is exactly S, the total space of the TM. A TM that runs in time T can have at most kT/l transitions, since for any head, the head must travel the l spaces between some two adjacent x_i's between any two transitions that are caused by this head. All together at most kTS/l bits were communicated in order to compute f thus this number is bounded from below by $C(f)$. The statement of the lemma follows. ∎

We thus obtain the following lower bound:

THEOREM 5. *For any fixed k, any $(k-1)$-head Turing machine computing the k-wise generalized inner product function on n bit strings requires a time–space trade-off of $TS \geqslant \Omega(n^2)$.*

Proof. Theorem 1 assures the high multiparty communication complexity of the generalized inner product function. In order to obtain the required spaces between any two adjacent x_i's in the input, we can restrict ourselves to the case where the first and last quarter of the bits in each x_i are fixed to zeroes (or any constants). This leaves the problem of computing the generalized inner product on $n/2$-bit long strings, with $n/2$ spaces between each two arguments, and the bound follows directly from the previous lemma. ∎

The bound given in Theorem 5 applies also to nondeterministic and to probabilistic (bounded two-sided error) Turing machines. This follows from Theorem 2′ (end of Section 2) and the fact that the simulation given in the proof of Lemma 4.1 carries over directly to allow probabilistic multiparty protocols to simulate probabilistic Turing machines (that compute f with bounded two-sided error) and nondeterministic protocols to simulate nondeterministic Turing machines.

4.2. *Trade-offs for Branching Programs and Formulas*

The cornerstone of the argument of Alon and Maass [AM] is a Ramsey-like lemma, which allowed them to use communication complexity lower bounds to obtain lower bounds for branching programs. The following generalization of that lemma will allow us to use multiparty communication complexity lower bounds to achieve improved lower bounds for branching programs.

DEFINITION. Let $\Sigma_1, \Sigma_2, ..., \Sigma_k$ be disjoint subsets of an alphabet Σ, and let $w \in \Sigma^*$. We say w has *t alternations relative to $\Sigma_1, ..., \Sigma_k$* if it can be broken up into $w = w_1 w_2 \cdots w_t$, where for each i there exists j_i such that $w_i \in (\Sigma - \Sigma_{j_i})^*$.

LEMMA 4.2. *Let $\Sigma_1, \Sigma_2, ..., \Sigma_k$ be disjoint subsets of Σ such that for each i, $|\Sigma_i| = n$. Let $w \in \Sigma^*$, where for every i, each letter in Σ_i appears at most l_i times in w, and let $l = \sum_i l_i$. Then it is possible to choose $\Pi_1 \subset \Sigma_1, ..., \Pi_k \subset \Sigma_k$, such that, for every i, $|\Pi_i| \geqslant n/3^{8l/k^2}$ and such that w has at most $\lceil 4l/k \rceil$ alternations relative to $\Pi_1, ..., \Pi_k$.*

Proof. The proof will proceed by induction on l. For $l < k$ the lemma is trivial since some $l_i = 0$. We shall assume the lemma is true for all integers smaller than l and prove it for l. During the proof we shall assume w is "very long," and thus every letter appears in it at some point; this will focus attention on the general case, and in the cases that w is "simpler" we can always imagine extending w to have the desired properties.

For a string u, we shall say *u uses Σ_i*, if at least $\frac{2}{3}$ of the letters $\sigma \in \Sigma_i$ appear in u. Let w_1 be the longest prefix of w that uses exactly $k-1$ of the Σ_i's. (Such a prefix must exist since the number of Σ_i's used can increase by at most one at every letter.) Let i_1 be the subscript of the "unused" Σ_i. Let w_2 be the longest prefix of $w - w_1$ (i.e., w, after the w_1 prefix is removed) that uses exactly $k-2$ of the $k-1$ Σ_i's used by w_1, and let i_2 be the unused index. We continue in this fashion obtaining a sequence $w_1, ..., w_{k/2}$ and $i_1, ..., i_{k/2}$, such that for $1 \leqslant j \leqslant k/2$ we have: (1) w_j does not use Σ_{i_j} and (2) for all i different from $i_1, ..., i_j$, w_j uses Σ_i (we do not care whether w_j uses $\Sigma_{i_1}, ..., \Sigma_{i_{j-1}}$ or not). Let w' be the suffix of w left after the removal of $w_1 \cdots w_{k/2}$.

For $j = 1, ..., k/2$, define Γ_{i_j} to be the subset of Σ_{i_j} of letters that did not appear in w_j. Note that $|\Gamma_{i_j}| \geqslant n/3$, and that relative to these Γ_{i_j}'s, each w_j has just one alternation.

For all i different from $i_1, ..., i_{k/2}$ define Γ_i to be the subset of Σ_i of letters that appeared in at least $k/4$ of the strings $w_1, ..., w_{k/2}$. We count the total number pairs (σ, j) such that $\sigma \in \Sigma_i$, $1 \leqslant j \leqslant k/2$, and σ appears in w_j. Since each letter in Γ_i appears in at most $k/2$ of the strings, and each letter in $\Sigma - \Gamma_i$ appears in at most $k/4$ of the strings, we obtain an upper bound of $(k/4) \cdot (n - |\Gamma_i|) + (k/2) \cdot |\Gamma_i|$. Since each of the strings $w_1, ..., w_{k/2}$ uses Σ_i we obtain a lower bound of $(2n/3) \cdot (k/2)$. A comparison of these two bounds implies that $|\Gamma_i| \geqslant n/3$.

Since for all i different from $i_1, ..., i_{k/2}$, each letter of Γ_i appeared at least $k/4$ times in the prefix $w_1 w_2 \cdots w_{k/2}$, we obtain that each such letter may appear at most $l_i - k/4$ times in w'. We are now ready to invoke the induction hypothesis on w' and $\Gamma_1, ..., \Gamma_k$, with n replaced by $n/3$ and l replaced by $l - k^2/8$ (since for $k/2$ values of i, l_i decreased by at least $k/4$). This gives subsets $\Pi_1, ..., \Pi_k$, each of size at most $(n/3)/3^{8(l-k^2/8)/k^2} = n/3^{8l/k^2}$, relative to which w' has $\lceil 4(l - k^2/8)/k \rceil$ alternations, to which we should add at most one alternation for each of $w_1, ..., w_{k/2}$. This adds up to at most $\lceil 4l/k \rceil$, as stated. ∎

This allows us to give width–length trade-offs for oblivious branching programs. Let f be a boolean function on k-tuples of n-bit binary strings with multiparty communication complexity $C(f)$ and let m be any integer. Define the function f^* as follows:

INPUT. k ternary strings each of length $2n3^m : x_1, ..., x_k \in \{0, 1, *\}^{2n3^m}$. Each ternary digit is encoded by two bits. (The total input size is $N = 4nk3^m$ bits).

OUTPUT. f computed on the strings derived after the deletion of the *'s from the input. (If the strings obtained after the deletion of the *'s are not all of length n, then output say, 0.)

LEMMA 4.3. *Any oblivious branching program computing f^* with length $L \leqslant Nkm/32$ has width $W \geqslant 2^{C(f)/km}$.*

Proof. We shall restrict the domain of f^* such as to achieve few alternations and still remain with the function f. First, for each $1 \leqslant i \leqslant k$ let l_i be the median of the number of times that each letter of x_i appears in the branching program (i.e., one of the two bits composing the ternary digit is queried by a level of the branching program). From the bound on the length of the branching program, an averaging argument shows that $\sum_i l_i \leqslant k^2m/8$. Our first restriction will be to substitute * for all the letters in x_i that appear more than l_i times in the branching program; this leaves $n3^m$ letters in each x_i.

We can now look at the string of input letters accessed by the branching program; this string is composed of letters belonging to the different x_i's. Viewed this way we now use the preceding lemma and conclude that there exist subsets of letters of each x_i, of size n, such that the branching program has at most km alternations relative to these sets. Let us restrict all the other input bits of f^* to *'s and look at the restricted branching program on these bits; this program computes f.

Note that k players in a multiparty communication protocol can simulate the branching program, with one player doing the simulation till an alternation occurs, and, on alternation, the name of the current vertex is sent to the next player. This requires $\log W$ bits of communication per each alternation of the branching program. The statement of the lemma follows, since the computation of f by simulating our restricted branching program will take $km \log W$ bits and that should be at least $C(f)$. ∎

Lemma 4.3, together with the lower bounds obtained in Theorem 1 on multiparty communication complexity, yields the following bound.

THEOREM 6. *Let f be the generalized inner product function, with $k = \log n/4$, and let f^* be as described previously for $m = \log n/4$, and let N be the total length of the input. Then any obliviuous branching program of length $o(N \log^2 N)$ requires width of $\exp(N^{\Omega(1)})$ to compute f^*.*

COROLLARY 4.4. *Any branching program computing f^* requires size $\Omega(N \log^2 N)$.*

COROLLARY 4.5. *Any boolean formula over an arbitrary finite basis computing f^* requires size $\Omega(N \log^2 N)$.*

Proof. Immediate from the following lemma. ∎

LEMMA 4.6. *A boolean formula of size l over a basis of arity d can be simulated by an oblivious branching program of length l and width l^d.*

Proof. By induction on the structure of the formula. Let the top gate of the formula be G, of arity k ($k \leqslant d$), and let the subformulas feeding to G be $f_1, ..., f_k$. We shall simulate the formula by simulating an oblivious decision tree for G, where each access to the value of some f_i will be replaced by recursively simulating f_i. The important thing to take care of is that the width does not grow too fast. This can be ensured by accessing the largest subformula first, then the second largest, etc. This way the largest subformula need only be simulated once, the second largest twice, and in general, the ith largest 2^{i-1} times.

The length of the resulting branching program is clearly the sum of the lengths of the subformulas. The size of the ith largest subformula is bounded from above by l/i, and the ith largest subformula is simulated at most 2^{i-1} times. Using the induction hypothesis we can therefore bound the width of the branching program by $\max_{i=1 \cdots d} 2^{i-1} l^d / i^d = l^d$. ∎

REFERENCES[1]

[AKLLR] R. ALELIUNAS, R. M. KARP, R. J. LIPTON, L. LOVÁSZ, AND C. RACKOFF, Random walks, universal traversing sequences and the complexity of maze problems, *in* "20th FOCS, 1979," pp. 218–223.

[AM] N. ALON AND W. MAASS, Meanders, Ramsey theory and lower bounds for branching programs, *in* "27th FOCS, 1986," pp. 410–417.

[AUY] A. V. AHO, J. D. ULLMANN, AND M. YANNAKAKIS, On notions of information transfer in VLSI circuits, *in* "15th STOC, 1983," pp. 133–139.

[AW] M. AJTAI AND A. WIGDERSON, Deterministic simulation of probabilistic constant depth circuits, *in* "26th FOCS, 1985," pp. 11–19.

[BCDRT] A. BORODIN, S. A. COOK, P. W. DYMOND, W. L. RUZZO, AND M. TOMPA, Two applications of complementation via inductive counting, "Proceedings, 3rd IEEE Symposium on Structure in Complexity Theory, 1988," pp. 116–125.

[BFS] L. BABAI, P. FRANKL, AND J. SIMON, Complexity classes in communication complexity theory, *in* "27th FOCS, 1986, pp. 337–347.

[BM] M. BLUM AND S. MICALI, How to generate cryptographically strong sequences of pseudo random bits, *in* "23rd FOCS, 1982," pp. 112–117.

[CFL] A. K. CHANDRA, M. L. FURST, AND R. J. LIPTON, Multiparty protocols, *in* "15th STOC, 1983," pp. 94–99.

[Co] A. COBHAM, "The Recognition Problem for the Set of Perfect Squares," Research Paper RC-1704, IBM, 1966.

[CG] B. CHOR AND O. GOLDREICH, Unbiased bits from sources of weak randomness and probabilistic communication complexity," *in* "26th FOCS, 1985," pp. 429–442.

[DG] P. DURIS AND Z. GALIL, A time-space tradeoff for language recognition, *Math. Systems Theory* **17** (1984), 3–12.

[ES] P. ERDÖS AND J. SPENCER, "Probabilistic Methods in Combinatorics," Academic Press, New York, 1974.

[1] FOCS = IEEE Symposium on Foundations of Computer Science. STOC = ACM Symposium on Theory of Computing.

[Gr] V. GROLMUSZ, The BNS lower bound for multi-party protocols is nearly optimal, *Inform. and Comput.*, to appear.

[GS] YU. GUREVICH AND S. SHELAH, Nondeterministic linear time tasks may require substantially nonlinear deterministic time in the case of sublinear work space, *in* "20th STOC, 1988," pp. 281–289.

[HG] J. HÅSTAD AND M. GOLDMANN, On the power of small depth threshold circuits, *in* "31st FOCS, 1990," pp. 610–618.

[ILL] R. IMPAGLIAZZO, L. LEVIN, AND M. LUBY, "Pseudorandom generation from one-way functions, *in* "21st STOC, 1989," pp. 12–24.

[Is] S. ISTRAIL, Polynomial universal traversing sequences for cycles are constructible, *in* "20th STOC, 1988," pp. 491–503.

[Ka] M. KARCHMER, Two time–space tradeoffs for element distinctness, *Theoret. Comput. Sci.* **47** (1986), 237–246.

[KLNS] J. D. KAHN, N. LINIAL, N. NISAN, AND M. E. SAKS, On the cover time of random walks in graphs, *J. Theoret. Probab.* **2** (1989), 121–128.

[Kn] D. E. KNUTH, "The Art of Computer Programming, Vol. II: Seminumerical Algorithms," Addison–Wesley, Reading, MA, 1981.

[KPS] H. J. KARLOFF, R. PATURI, AND J. SIMON, "Universal traversal sequences of length $n^{O(\log n)}$ for cliques," *Inform. Process. Lett.* **28** (1988), 241–243.

[LVW] M. LUBY, B. VELIČKOVIĆ, AND A. WIGDERSON, Deterministic approximate counting for multivariate polynomials over GF[2], in preparation.

[Ne] E. I. NEČIPORUK, A boolean function, *Sov. Math. Dokl.* **7**, No. 4 (1966), 999–1000.

[Ni1] N. NISAN, On read-once vs. multiple access to randomness in Logspace, *in* "Proceedings, 5th IEEE Structure in Complexity Theory Conference Barcelona, 1990," pp. 179–184.

[Ni2] N. NISAN, Pseudorandom generators for space-bounded computation, *in* "22nd STOC, Baltimore, 1990," pp. 204–212.

[Ni3] N. NISAN, Pseudorandom bits for constant depth circuits, *Combinatorica* **11** (1991), 63–70.

[NW] N. NISAN AND A. WIGDERSON, Hardness vs. randomness, *in* "29th FOCS, 1988," pp. 2–11.

[RT] J. H. REIF AND J. D. TYGAR, "Towards a Theory of Parallel Randomized Computation," TR-07-84, Aiken Computation Laboratory, Harvard University, 1984.

[Sch] W. M. SCHMIDT, "Equations over Finite Fields, An Elementary Approach," Lecture Notes in Mathematics, Vol. 536, Springer-Verlag, Berlin/New York, 1976.

[Va] U. V. VAZIRANI, Strong communication complexity or generating quasi-random sequences from two communicating semirandom sources, *Combinatorica* **7** (1987), 375–392.

[Y1] A. C. YAO, Some complexity questions related to distributive computing, *in* "11th STOC, 1979," pp. 209–213.

[Y2] A. C. YAO, Theory and applications of trapdoor functions, *in* "23rd FOCS, 1982," pp. 80–91.

[Y3] A. C. YAO, Lower bounds by probabilistic arguments, *in* "24th FOCS, 1984, pp. 420–428.

[Y4] A. C. YAO, "On ACC and threshold circuits," *in* "31th FOCS, 1990," pp 619–627.