Electronic Colloquium on Computational Complexity, Report No. 79 (2005)

Expanders and time-restricted branching programs

S. Jukna *†‡

Abstract

The *replication number* of a branching program is the minimum number R such that along every accepting computation at most R variables are tested more than once. Hence $0 \leq R \leq n$ for every branching program in n variables. The best results so far were exponential lower bounds on the size of branching programs with $R = o(n/\log n)$. We improve this to $R = \epsilon n$ for a constant $\epsilon > 0$. This also gives a new and simple proof of an exponential lower bound for branching programs of length $(1+\epsilon)n$. These lower bounds are proved for quadratic functions of Ramanujan graphs.

1 Introduction

We consider the standard model of (deterministic) branching programs (see, e.g. the survey [16] or the monograph [20]). Recall that such a program is just a directed acyclic graph with one source node. Each sink (i.e. a node of outdegree 0) is labeled either by 1 (accept) or by 0 (reject). Each non-sink node has outdegree 2, and the (two) outgoing edges are labeled by the tests $x_i = 0$ and $x_i = 1$, for some $i \in \{1, \ldots, n\}$. Such a program computes a boolean function $f : \{0,1\}^n \to \{0,1\}$ in a natural way: given an input vector $x \in \{0,1\}^n$ we start in the source node, and follow the (unique) path whose tests are consistent with the corresponding bits of x; this path is the *computation* on x. This way we reach a sink, and the input x is accepted iff this is the 1-sink.

Natural parameters of every branching program are:

- the *size* S = the number of nodes;
- the *computation time* T = the length of a longest computation, and

*Institute of Mathematics and Informatics, Vilnius, Lithuania

†*Current Address*: Universität Frankfurt, Institut für Informatik, Robert-Mayer-Str. 11-15, D-60054 Frankfurt, Germany. Email: jukna@thi.informatik.uni-frankfurt.de

‡Research supported by a DFG grant SCHN 503/2-2.

ISSN 1433-8092

- the *replication number* $R =$ the maximal number of bits tested more than once along an accepting computation.

Note that for every branching program in n variables we have $0 \le R \le n$. Moreover, every boolean function f in n variables can be computed by a branching program with $T = n$ and $R = 0$: just take a complete binary tree of depth n. However, the size S of such (trivial) branching programs is then exponential for most functions. It is therefore interesting to understand whether S can be substantially reduced by allowing larger values for T and/or R. This is a so-called "space versus time" problem for branching programs.

Thus, given a boolean function f in n variables, we are interested in the smallest size S of a branching program computing f when either T or R (or both) are limited.

Note that T and R are "semantic" restrictions: they concern only *consistent* paths (computations), i.e. paths that do not contain two contradicting tests $x_i = 0$ and $x_i = 1$ on some bit i. The "syntactic" case, where the restriction is on *all* paths (be they consistent or not) is usually easier to deal with, and exponential lower bounds on the size S in this case were obtained for $T = o(n \log n)$ [15, 6, 10] as well as for[1] $R = o(n^{1/3}/\log^{2/3} n)$ [19, 18].

In the non-syntactic case, the first super-polynomial lower bounds on S with $R = o(n/(\log n)^3)$ were proved in [17] (improving upon [21]); this was further improved to $R = o(n/\log n)$ in [11]. These bounds hold also for $T = (1+\epsilon)n$ with $\epsilon = o(1/\log n)$.

The first exponential lower bound on S for $T = (1+\epsilon)n$ with a (very small but constant!) $\epsilon > 0$ was proved in [4] (the proof works for $\epsilon = 0,0178$). Shortly after this was substantially improved in [1] to $T = cn$ for an arbitrary constant $c > 0$ (see also [5] for some further improvements of this result).

In this paper we prove exponential lower bounds on the size S when $R = \epsilon n$ for a constant $\epsilon > 0$ (Theorem 4.1 below). This improves the lower bounds of [21, 17, 11] and gives a new proof for the lower bound of [4] (for a different function). The amazing simplicity of our proofs (modulo some known deep constructions of expander graphs) indicates that expander graphs could be good candidates to construct hard boolean functions for time-restricted branching programs.

We prove our lower bounds for quadratic forms $f(x) = x^{\mathrm{T}} A x$ over $GF(2)$ where A is an adjacency matrix of particular Ramanujan graphs. It should be noted that quadratic forms (over different fields) were used in most papers on time-restricted branching programs: Sylvester and generalized Fourier matrices in [6, 4, 5], Hankel matrices in [1, 5], etc. The "hardness" of the resulting functions was achieved by special *algebraic* properties of the underlying matrices A: every large enough submatrix must have large rank. The difference of our proof is that we use the *combinatorial* properties of the underlying matrices A: they

[1] In the literature, branching programs with the replication number R are also called "$(1, +R)$-branching programs."

must have relatively few 1's and still do not have large all-0 submatrices. Such are, in particular, adjacency matrices of good expander graphs, including the Ramanujan graphs.

2 A general lower bound

A set of vectors $A \subseteq \{0,1\}^n$ is a (boolean) *rectangle* if there is a partition of $\{1,\ldots,n\}$ into two disjoint parts X_1 and X_2 of the same size ± 1 (i.e. a partition must be balanced) and subsets $A_i \subseteq \{0,1\}^{X_i}$ such that $A = A_1 \times A_2$. That is, the characteristic function f_A of A ($f_A(x) = 1$ iff $x \in A$) can be represented as an AND $f_A = f_1(X_1) \wedge f_2(X_2)$ of two boolean functions on the corresponding sets of variables. The *width* of a rectangle $A = A_1 \times A_2$ is $\min\{|A_1|, |A_2|\}$. Define the *rectangle width* $w(f)$ of a boolean function f as the maximal possible width of a rectangle A such that $f(x) = 1$ for all $x \in A$ (in this case we also say that the rectangle is *contained* in f).

We say that a boolean function $f : \{0,1\}^n \to \{0,1\}$

- is *dense* if $|f^{-1}(1)| \geq 2^{n-o(n)}$;
- has *small* rectangle width if $w(f) \leq 2^{n/2-\delta n}$ for some constant $\delta > 0$;
- is *good* if any two vectors in $f^{-1}(1)$ differ in at least two bits.

Theorem 2.1. *Let f be a good and dense boolean function in n variables. If f has small rectangle width then there is a constant $\epsilon > 0$ such that any deterministic branching program computing f with the replication number $R = \epsilon n$ has size $S = 2^{\Omega(n)}$.*

We postpone the (relatively simple) proof of this theorem to Section 5, and turn to its applications.

3 Functions with small rectangle width

To apply Theorem 2.1 we need boolean functions $f : \{0,1\}^n \to \{0,1\}$ for which $w(f)$ is much smaller than the trivial upper bound $2^{n/2}$. Recall that $w(f)$ is the maximum over *all* balanced partitions. This gives an adversary a lot of freedom to enforce a wide rectangle. We must therefore choose the function f so that no partition is suitable (for the adversary). We define such functions using graphs.

Let $G = (V, E)$ be an undirected graph, and A its adjacency matrix. By f_G we denote the quadratic form $f_G(x) = x^{\mathrm{T}} Ax$ over $GF(2)$. That is,

$$f_G(x) = \sum_{uv \in E} x_u x_v \bmod 2.$$

Let us first show that such functions are dense.

Proposition 3.1. *For every graph G with at least one edge the function f_G is dense.*

Proof. Take an arbitrary edge uv of $G = (V, E)$ and consider the induced subgraph H of G on the vertex set $V \setminus \{u, v\}$. Given an assignment $a \in \{0,1\}^{V\setminus\{u,v\}}$ of constants to the vertices of H, let $p_u(a)$ and $p_v(a)$ be the parities of bits assigned to the neighbors of u and v in H. Then

$$f_G(a, x_u, x_v) = f_H(a) \oplus x_u x_v \oplus p_u(a)x_u \oplus p_v(a)x_v.$$

If $f_H(a) = 1$ then set $x_u = x_v = 0$. If $f_H(a) = 0$ and $p_u(a) = p_v(a)$ then set $x_u = x_v = 1$. If $f_H(a) = 0$ and $p_u(a) \neq p_v(a)$ with, say, $p_u(a) = 1$ then set $x_u = 1$ and $x_v = 0$. It is easy to see that in all three cases the resulting assignment will be accepted by f_G. Hence, at least one extension of *every* assignment $a \in \{0,1\}^{V\setminus\{u,v\}}$ will be accepted by f_G, implying that $|f_G^{-1}(1)| \geq 2^{n-2}$. □

A more interesting fact is that quadratic forms of some graphs have small rectangle width.

Say that a graph is *s-mixed* if every pair of disjoint sets of s vertices is joined by at least one edge, that is, the complement of the graph contains no copies of $K_{s,s}$; the smaller s is, the more is the graph mixed. The following lemma (inspired by the paper of Hayes [8]) says that if the graph G has constant degree and is still mixed enough, then f_G has small rectangle width.

Lemma 3.2. *Let G be an n-vertex graph of maximum degree d. If G is s-mixed then $w(f_G) \leq 2^{n/2-\delta n+1}$ where $\delta = \frac{1}{8d}\left(1 - \frac{2s}{n}\right)$.*

Proof. Let $g(X) = g_1(X_1) \wedge g_2(X_2)$ be an arbitrary rectangle contained in f_G; hence $n/2 \leq |X_i| \leq n/2 + 1$ for both $i = 1, 2$. Recall that variables in X correspond to vertices of G, and construct an induced matching $M = \{x_1y_1, \dots, x_my_m\}$ from X_1 to X_2 by repeatedly taking an edge $xy \in E$ with $x \in X_1$ and $y \in X_2$ and removing it together with all its neighbors. Since the graph G is s-mixed and both parts X_1 and X_2 have at least $n/2$ vertices, the procedure will run as long as $n/2 - 2dm \geq s$. Hence, $m \geq (n/2 - s)/(2d) = 2\delta n$.

We now set to 0 all variables corresponding to vertices outside the matching M. Since M is an *induced* subgraph of G, the obtained (from f_G) function is just the inner product function

$$f_M(x_1, \dots, x_m, y_1, \dots, y_m) = \sum_{i=1}^{m} x_i y_i \bmod 2.$$

Moreover, the obtained (from g) function $g'(X') = g_1'(X_1') \wedge g_2'(X_2')$ is a rectangle (on $|X'| = 2m$ variables) with $X_1' = \{x_1, \dots, x_m\}$ and $X_2' = \{y_1, \dots, y_m\}$, and is contained in f_M (because $g(X)$ was contained in f_G). Lindsey's Lemma (see, e.g. [7]) implies that for every two subsets A, B of $\{0,1\}^m$,

$$\left| \sum_{a\in A, b\in B} (-1)^{f_M(a,b)} \right| \leq 2^{m/2}\sqrt{|A| \cdot |B|}.$$

In particular, f_M can be constant on $A \times B$ only if $|A| \cdot |B| \leq 2^m$. Hence, the rectangle g' cannot have width larger than $2^{m/2}$. Since at most $|X_i| - m \leq n/2 + 1 - m$ variables in each block of the partition $X = X_1 \cup X_2$ were set to 0, the original rectangle can have width at most $2^{m/2} \cdot 2^{n/2+1-m} = 2^{n/2+1-m/2} \leq 2^{n/2-\delta n+1}$. $\square$

According to Lemma 3.2 we need explicit constant-degree graphs with good expansion properties. For this purpose we take Ramanujan graphs, i.e. $(q+1)$-regular graphs with the property that $|\lambda| \leq 2\sqrt{q}$ for every nontrivial (i.e. $\neq \pm(q+1)$) eigenvalue of their adjacency matrix, this is almost optimal because $\lambda \geq 2\sqrt{q} - o(1)$ for any $(q+1)$-regular graph [2]. Explicit constructions of Ramanujan graphs on n vertices for every prime $q \equiv 1 \bmod 4$ (and infinitely many values of n) were given in [13, 12]; these were later extended to the case where q is an arbitrary prime power in [14, 9].

The so-called Expander Mixing Lemma (see, e.g., Corollary 9.2.5 in [3]) states that if G is a d-regular graph on n vertices and λ is the second largest eigenvalue of its adjacency matrix, then the number $e(A, B)$ of edges between every two (not necessarily disjoint) subsets A and B of vertices satisfies

$$\left| e(A, B) - \frac{d}{n}|A| \cdot |B| \right| \leq \lambda \sqrt{|A| \cdot |B|}.$$

According to this lemma, $(q+1)$-regular Ramanujan graphs are s-mixed for $s = 2n/\sqrt{q}$.

4 A lower bound for an explicit function

Let $q > 16$ be a prime power, and let $G = (V, E)$ be a $(q+1)$-regular Ramanujan graph on n vertices. By taking the AND with the parity function $\mathrm{Parity}(X) = \sum_{u \in V} x_u \bmod 2$ we ensure that the resulting function

$$\mathsf{f}_n(X) = f_G(X) \wedge \mathrm{Parity}(X)$$

is good.

Theorem 4.1. *There is a constant $\epsilon > 0$ such that any deterministic branching program computing f_n with the replication number $R = \epsilon n$ requires size $2^{\Omega(n)}$.*

Proof. By Proposition 3.1, the function f_G (and hence, also f_n) is a dense boolean function. On the other hand, since G is s-mixed for $s = 2n/\sqrt{q}$, Lemma 3.2 implies that $w(\mathsf{f}_n) \leq w(f_G) \leq 2^{n/2-\delta n+1}$ where

$$\delta = \frac{1}{8(q+1)}\left(1 - \frac{2s}{n}\right) \geq \frac{1}{8(q+1)}\left(1 - \frac{4}{\sqrt{q}}\right)$$

is a positive constant, as long as $q > 16$. It remains to apply Theorem 2.1. $\square$

If a branching program computes a good boolean function in n variables and has length (time) T, then its replication number R cannot exceed $T - n$. Hence, Theorem 4.1 yields an exponential lower bound also for the class of time $(1+\epsilon)n$ branching programs for a constant $\epsilon > 0$.

5 Proof of Theorem 2.1

We prove the following more general fact.

Theorem 5.1. *Let f be a good and dense boolean function in n variables, and let S be the minimum size of a deterministic branching program with replication number R computing f. Then*

$$\log_2 S \geq \log_2 |f^{-1}(1)| - \frac{n}{2} - \log_2 w(f) - R \log_2 \frac{2en}{R} - 3.$$

Theorem 2.1 is a direct consequence of this fact: if $w(f) \leq 2^{n/2-\delta n}$ for some constant $\delta > 0$, then it is enough to take $R = \epsilon n$ for a constant $\epsilon > 0$ such that $\epsilon \log_2(2e/\epsilon) < \delta$.

Proof. Take an arbitrary deterministic branching program computing f. Let S be the size and R be the replication number of this program. For an input $x \in f^{-1}(1)$, let $comp(x)$ denote the (accepting) computation path on x. Since f is good, all n bits are tested at least once along each of these paths. Split each of the paths $comp(x)$ into two parts $comp(x) = (p_x, q_x)$ where p_x is an initial segment of $comp(x)$ along which $n/2$ different bits are tested. Hence, the remaining part q_x can test at most $n/2 + R + 1$ different bits.[2]

If we replace each test $x_i = 1$ by the variable x_i and each test $x_i = 0$ by its negation $\neg x_i$, then we can look at segments p_x and q_x as monoms, i.e. as ANDs of corresponding variables or their negations. This way we obtain that f can be written as an OR of at most S ANDs $D_1 \wedge D_2$ of DNFs such that

(i) All monoms have length at most $k = n/2 + R + 1$ and at least $n/2$. This holds by the choice of segments p_x and q_x.

(ii) Any two monoms in each DNF are inconsistent, i.e. one contains a variable x_i and the other contains its negation $\neg x_i$. This holds because the program is deterministic: the paths must split before they meet.

(iii) For all $p_1 \in D_1$ and $p_2 \in D_2$ either $p_1 p_2 = 0$ or $|X(p_1) \cap X(p_2)| \leq R$ where $X(p)$ is the set of variables in monom p. This holds because the program has replication number R.

[2]Note that we count only the number of tests of *different* bits—the total length of (the number of tests along) $comp(x)$ may be much larger than $n + R$.

By (i), each monom of D_i accepts at least a 2^{-k} fraction of all 2^n vectors and, by (ii), no two of them accept the same vector. Hence, each of the DNFs D_1 and D_2 can have at most 2^k monoms.

Fix now one AND $g = D_1 \wedge D_2$ for which the set $B = g^{-1}(1)$ is the largest one; hence, $S \geq |f^{-1}(1)|/|B|$. To finish the proof of the theorem, it is enough to show that B contains a rectangle of width (recall that $k = n/2 + R + 1$)

$$w \geq \frac{1}{3} \cdot \frac{|B|}{2^k \left(\frac{en}{R}\right)^R}.$$

For every $a \in B$ there are monoms $p_i \in D_i$ such that $p_1 p_2(a) = 1$. The (potential) problem, however, is that for different vectors a the corresponding monoms p_1 and p_2 may share *different* variables in common. This may prohibit their combination into a rectangle. To avoid this problem, we use the last property (iii) and fix a set Y of $|Y| \leq R$ variables for which the set

$$A = \{a \in B : \exists p_i \in D_i : \ p_1 p_2(a) = 1 \text{ and } X(p_1) \cap X(p_2) = Y\}$$

is the largest one. Hence,

$$|A| \geq |B| \Big/ \sum_{i=0}^{R} \binom{n}{i} \geq |B| \cdot \left(\frac{en}{R}\right)^{-R}.$$

Each $a \in A$ is an extension of some monom $p \in D_1$, i.e. $pq(a) = 1$ for some $p \in D_1$ and $q \in D_2$ with $X(p) \cap X(q) = Y$. Since the monoms of D_1 are mutually inconsistent, no two of them can have a common extension. Hence, the sets

$$\mathrm{ext}_A(p) = \{a \in A : \exists q \in D_2 : \ pq(a) = 1 \text{ and } X(p) \cap X(q) = Y\}$$

with $p \in D_1$ form a partition of A into $|D_1|$ blocks. Since the average size of a block is $|A|/|D_1| \geq |A|/2^k$, at least a $\frac{2}{3}$ fraction of all vectors of A must belong to blocks of size at least $\frac{1}{3} \cdot |A|/2^k$. The same holds also for the partition of A given by the extensions of the monoms from the second DNF D_2. Hence, if we set

$$A_i = \{a \in A : \ p(a) = 1 \text{ for some } p \in D_i \text{ with } |\mathrm{ext}_A(p)| \geq \tfrac{1}{3} \cdot |A|/2^k\},$$

then both A_1 and A_2 consist of at least $\frac{2}{3}|A|$ vectors of A, implying that $A_1 \cap A_2 \neq \emptyset$.

Fix a vector $a \in A_1 \cap A_2$ and let $p_1 \in D_1$ and $p_2 \in D_2$ be the corresponding monoms for which $p_i(a) = 1$, $|\mathrm{ext}_A(p_i)| \geq \frac{1}{3}|A|/2^k$ and $X(p_1) \cap X(p_2) = Y$. The combined monom $p_1 p_2$ is consistent (with a) and contains all n variables (since A is good).

Let b be the projection of the vector a onto Y. All the vectors in $\mathrm{ext}_A(p_1)$ and in $\mathrm{ext}_A(p_2)$ coincide with b on Y. Consider the rectangle $C = C_1 \times \{b\} \times C_2$ where C_i is the projection of

$\mathrm{ext}_A(p_{3-i})$ onto the set of variables $X(p_i) \setminus S$. (This is a rectangle, because both $|X(p_1)|$ and $|X(p_2)|$ are at least $n/2$.) Hence, we have a rectangle of width $|C_i| = |\mathrm{ext}_A(p_{3-i})| \geq \frac{1}{3}|A|/2^k$, and it remains to show that all the vectors of this rectangle are accepted by $g = D_1 \wedge D_2$.

Claim 5.2. *$g(c) = 1$ for all $c \in C$.*

Proof. The vector a belongs to C and has the form $a = (a_1, b, a_2)$ with $a_i \in \{0,1\}^{X(p_i)\setminus S}$. Take now an arbitrary vector $c = (c_1, b, c_2)$ in C.

The vector (a_1, b, c_2) is in $\mathrm{ext}_A(p_1)$. Hence, there must be a monom $q_2 \in D_2$ such that $p_1 q_2$ accepts this vector and $X(p_1) \cap X(q_2) = Y$. Since all bits of a_1 are tested in p_1 and none of them belongs to Y, none of these bits is tested in q_2. Hence, q_2 must accept also the vector $c = (c_1, b, c_2)$. Similarly, using the fact that (c_1, b, a_2) is in $\mathrm{ext}_A(p_2)$, we can conclude that the vector $c = (c_1, b, c_2)$ is accepted by some monom $p_2 \in D_1$. Hence, the vector c is accepted by both DNFs D_1 and D_2, as desired.

This completes the proof of the claim, and thus, the proof of the theorem. ☐

Acknowledgments I would like to thank Martin Sauerhoff and Detlef Sieling for useful comments.

References

[1] Ajtai, M. (1999): A non-linear time lower bound for boolean branching programs. In *Proc. of 40th Ann. IEEE Symp. on Foundations of Comput. Sci.*, 60–70.

[2] Alon, N. (1986): Eigenvalues and expanders, *Combinatorica*, **6**, 83–96.

[3] Alon, N. and Spencer, J. H. (2000): *The probabilistic method.* 2nd ed. Wiley.

[4] Beame, P, Jayram, T. S.[3] and Saks, M. (2001): Time-space tradeoffs for branching programs, *J. Comput. Syst. Sci.* **63**: 4, 542–572. Preliminary version in: *Proc. of 39th Ann. IEEE Symp. on Foundations of Comput. Sci.* (1998), 254–263

[5] Beame, P. Saks, M. Sun, X. and Vee, E. (2003): Time-space trade-off lower bounds for randomized computation of decision problems. *Journal of the ACM*, **50**:2,154–195.

[6] Borodin, A., Razborov, A. and Smolensky, R. (1993): On lower bounds for read-k times branching programs, *Computational Complexity* **3**, 1–18.

[7] Erdős, P. and Spencer, J. (1974): *Probabilistic methods in combinatorics.* Budapest: Akademiai Kiado.

[8] Hayes, T. P. (2003): Separating the k-party communication complexity hierarchy: an application of the Zarankiewicz problem. Manuskript.

[9] Jordan, B. W. and Livné, R. (1997): Ramanujan local systems on graphs, *Topology*, **36**:5, 1007–1024.

[3] Formely Jayram S. Thathachar

[10] Jukna, S. (1995): A note on read-k-times branching programs, *Informatique theorique et Applications/Theoretical Informatics and Applications* **29**:1, 75–83.

[11] Jukna, S. and Razborov,A. (1998): Neither reading few bits twice nor reading illegally helps much, *Discrete Appl. Math.*, **85**, pp. 223–238.

[12] Lubotzky, A. Phillips, R. and Sarnak, P. (1988): Ramanujan graphs, *Combinatorica* **8**:3, 261–277.

[13] Margulis, G. A. (1973): Explicit constructions of concentrators, *Probl. Peredachi Inf.* **9**, 71–80. English translation in *Problems of Inf. Transm.* (1975), 323–332.

[14] Morgenstern, M. (1994): Existence and explicit constructions of $q+1$ regular Ramanujan graphs for every prime power q, *J. Comb. Theory Ser. B*, **62**:1, 44–62.

[15] Okolnishnikova, E. A. (1991): Lower bounds for branching programs computing characteristic functions of binary codes, *Metody discretnogo analiza* **51** 61–83 (in Russian).

[16] Razborov, A. A. (1991): Lower bounds for deterministic and nondeterministic branching programs. In: Springer Lecture Notes in Computer Science, vol. 529, 47–60.

[17] Savický, P. and Zák, S. (1997): A lower bound on branching programs reading some bits twice, *Theor. Comput. Sci.* **172**, No.1-2, 293–301.

[18] Sieling, D. (1996): New lower bounds and hierarchy results for restricted branching programs,*J. Comput. Syst. Sci.* **53**:1 79–87.

[19] Sieling, D. and Wegener, I. (1994): New lower bounds and hierarchy results for restricted branching programs. In: Springer Lecture Notes in Computer Science, vol. 903, 359–370.

[20] Wegener, I. (2000): *Branching programs and Binary Decision Diagrams: Theory and Applications.* SIAM Series in Discrete Mathematics and Applications.

[21] Žák, S. (1995): A superpolynomial lower bound for $(1, +k(n))$-branching programs. In: Springer Lecture Notes in Computer Science, vol. 969, 319–325.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'