Electronic Colloquium on Computational Complexity - Reports Series 1996 - available via:

Revision 01 of
ECCC TR96-036

FTP: ftp.eccc.uni-trier.de:/pub/eccc/
WWW: http://www.eccc.uni-trier.de/eccc/
Email: ftpmail@ftp.eccc.uni-trier.de with subject 'help eccc'

A large lower bound for 1-branching programs

Petr Savický, Stanislav Žák*
Institute of Computer Science, Academy of Sciences of Czech Republic,
Pod vodárenskou věží 2, 182 07 Praha 8, Czech Republic
e-mail: {savicky,stan}@uivt.cas.cz

Abstract

Branching programs (b. p.'s) or decision diagrams are a general graph-based model of sequential computation. B.p.'s of polynomial size are a nonuniform counterpart of LOG. Lower bounds for different kinds of restricted b. p.'s are intensively investigated. An important restriction are the so called 1–b. p.'s, where each computation reads each input bit at most once. There is a series of lower bounds for 1–b. p.'s. The largest known lower bound was $2^{n/2000}$ for a function of n variables, see [14]. In the present paper, a lower bound of $2^{n-3\sqrt{n}}$ is given for an explicit function. A generalization of the construction is also presented that may in principle lead to a lower bound that almost matches a general upper bound.

1 Introduction

A branching program (b. p.) is a computation model for representing Boolean functions. The input of a branching program is a vector consisting of n input bits. The branching program itself is a directed acyclic graph with one source. The out-degree of each node is at most 2. Every branching node, i.e. a node of out-degree 2, is labeled by an index of an input bit and one of its out-going edges is labeled by 0, the other one by 1. The sinks (out-degree 0) are labeled by 0 and 1. A branching program determines a Boolean function as follows. For an input vector the computation starts at the source. If a node of out-degree 1 is reached, the computation follows the unique edge leaving the node. In each branching node the input bit, assigned to the node, is tested and the out-going edge, labeled by the actual value of the input bit, is chosen. Finaly, a sink is reached. Its label determines the value of the function for the given input. By the size of a branching program we mean the number of its nodes.

Branching programs are a model of the configuration space of Turing machines, where each node corresponds to a configuration. Thus the polynomial size b. p.'s represent a nonuniform variant of LOG. A superpolynomial lower bound on b. p.'s for a Boolean function computable within polynomial time would imply $P \neq LOG$.

In order to investigate the computing power of branching programs, restricted models were suggested. One of the most basic restrictions are read once branching programs (1–b. p.), where the restriction is such that during each computation each input bit is tested at most once.

There is a series of lower bounds for 1–b. p.'s. For a Boolean function called half-clique only a lower bound $2^{\Omega(\sqrt{n})}$, where n is the number of input bits, was proved in [19]. Independently, for the clique function with appropriate parameters, a lower bound $2^{\Omega(\sqrt{n})}$ was proved

*The research of both authors was supported by GA of the Czech Republic, grant No. 201/95/0976.

in [17]. A lower bound $2^{\sqrt{n/2}-O(1)}$ appeared in [5] for testing existence of a Hamiltonian cycle and existence of a perfect matching in a graph. In [2] a lower bound $2^{n/c}$ for a large c is proved for the function "parity of the number of triangles in a graph". This lower bound was improved in [14] to a lower bound $2^{n/2000}$ for the same function. In [14] a technique for proving lower bounds for 1–b. p.'s that generalizes most of the previous results on 1-b. p.'s, is described. In [10], a bound $2^{\Omega(\sqrt{n})}$ is proved for multiplication.

Besides 1–b. p.'s more general models are investigated. Namely, k–b. p.'s and syntactic k–b. p.'s. In a k–b. p., every computation is allowed to test each input bit at most k times. Syntactic k–b. p.'s are k–b. p.'s, where the restriction of at most k tests of each input bit is applied not only to valid computations, but to all paths from the source to a sink. Note that each 1–b. p. is a syntactic 1–b. p., while for $k \geq 2$ this is not true in general. For syntactic k–b. p.'s for k up to $\Omega(\log n)$, exponential lower bounds are known, see [3], [6], [9]. The results [3], [6] apply even to nondeterministic syntactic k–b. p.'s. For general (nonsyntactic) k–b. p.'s, proving exponential lower bounds for explicit functions is an open problem even for $k = 2$.

Another generalization of 1–b. p.'s are $(1, +k)$–b. p.'s. In a $(1, +k)$–b. p., in any computation, at most k of the input bits may be tested more than once. For such b. p.'s, exponential lower bounds are known, see [7], [11] and [20].

The 1–b. p.'s are also interesting as a data structure for representing Boolean functions. If a function f is represented by a small 1–b. p., it is possible to test if the function f is not a zero function (satisfiability test) efficiently. It is also possible to count the number of assignments x satisfying $f(x) = 1$. Both these problems may be solved in time polynomial in the size of the representation. Moreover, there is a probabilistic test of equivalence of two 1–b. p.'s, working in time polynomial in the size of the input b. p.'s. Since for some other operations there are no efficient algorithms, even more restricted models are used, namely OBDDs (ordered binary decision diagrams). OBDDs are used e.g. for representing the Boolean functions in some CAD applications like the design or verification of Boolean circuits. As a review paper see [18].

In the present paper, a lower bound $2^{n-3\sqrt{n}}$ on the size of 1–b. p.'s is proved for an explicit function of n variables. The function is constructed in the form $f(x) = x_{\phi(x)}$, where $\phi : \{0,1\}^n \to \{1, 2, \ldots, n\}$ is a specific function. In the last section we present a condition on the mapping ϕ that generalizes the properties of ϕ used to prove the lower bound for $x_{\phi(x)}$ in Section 2. The condition implies a lower bound $2^{n-O(\log n)}$ and we provide a nonconstructive proof of the fact that functions satisfying the condition exist. An explicit construction of such a mapping ϕ would imply a lower bound that almost matches a general upper bound on the size of 1–b. p.'s that is $2^{n-\log n+O(1)}$.

2 The lower bound

We shall work with Boolean functions of n variables. By a partial input we understand an element of $\{0, 1, *\}^n$. As usual, the positions containing 0 or 1 mean that the corresponding input bit is fixed to the specified value, while a $*$ means that the input bit remains free. We say that a partial input u is defined on I, if $u_i \in \{0, 1\}$ for all $i \in I$ and $u_i = *$ otherwise.

For a subset $I \subseteq \{1, 2, \ldots, n\}$, let $\mathcal{B}(I)$ mean the set of partial inputs defined on I. Let f be a Boolean function, I a set of indices of the variables and let u be a partial input from $\mathcal{B}(I)$. Then, let $f|_u$ mean the subfunction of f obtained from f by setting x_i to u_i for all $i \in I$. Let I and J be disjoint sets of indices of bits and let $u \in \mathcal{B}(I)$ and $v \in \mathcal{B}(J)$. Then, let

$[u,v]$ mean the input from $\mathcal{B}(I \cup J)$ in which the bits with indices from I resp. J have the same values as in u resp. v.

To prove lower bounds on 1–b. p.'s, we shall use the technique from [14]. For a Boolean function f of n variables and a subset $I \subseteq \{1,2,\ldots,n\}$, let

$$\nu(f,I) =_{df} \max_{u \in \mathcal{B}(I)} |\{v \in \mathcal{B}(I) : f|_v = f|_u\}| .$$

The following theorem (Theorem 2.1 from [14]) is a special case of a general lower bound for 1-b. p.'s also presented in [14]. For convenience of the reader, we include a complete proof of the special case.

Theorem 2.1 (Simon, Szegedy) *Let f be a Boolean function of n variables and let $r \leq n$ be an integer. Then, any 1–b. p. computing f has size at least*

$$\frac{2^{n-r}}{\max\limits_{|I|=n-r} \nu(f,I)}.$$

Proof: As in [14], for any node v of a b. p., let v^+ be the set of indices of input bits read in v or on some directed path starting in v.

Let P be a 1–b. p. computing a function f. For every input x, let $e(x)$ be the edge (u,v) such that both u and v belong to the computation path for x and $|u^+| \geq r+1$ and $|v^+| \leq r$. For each x, such an edge exists and it is unique.

Fix some input x and let (u,v) be the edge $e(x)$. We are going to bound from above the number of inputs y satisfying $e(y) = e(x)$.

Let u be labeled by x_i. Let J be a set of size r such that $u^+ \setminus \{i\} \supseteq J \supseteq v^+$. Let $I = \{1,2,\ldots,n\} \setminus J$, especially $i \in I$. For every input y, let y^* be the restriction of y to the indices from I. Note that, for every y going through the edge (u,v), all the bits read before reaching v are preserved in y^*. Hence, for every y, the computation for y goes through (u,v) if and only if the computation for y^* goes through (u,v).

If $e(y) = e(x)$ then the computation for both y^* and x^* leads to the node v. Hence, for every partial input z defined on J, we have $f([y^*,z]) = f([x^*,z])$. In other words, $f|_{y^*} = f|_{x^*}$. There are at most $\nu(f,I)$ different y^* for inputs y satisfying this. Since every y^* is a restriction of 2^r total inputs y, there are at most $2^r \nu(f,I)$ inputs y satisfying $e(y) = e(x)$.

The upper bound from the previous paragraph is valid for any input x and the corresponding set I of size $n-r$. Hence, the number of edges in the b. p. is at least $2^{n-r} / \max_{|I|=n-r} \nu(f,I)$. This implies that the size of P is at least one half of this number, since the out-degree of every node is at most 2.

In fact, one can improve the lower bound for the number of nodes of P by a factor of two using the following. The set of edges $e(x)$ for all $x \in \{0,1\}^n$ has the property that no two of these edges participate in a directed path. An easy argument shows that if G is a directed acyclic rooted graph such that the out-degree of each node is at most two and S is a subset of the edges with the above property, then $|S| \leq |G|$. □

In order to define the Boolean function f_n, for which the lower bound will be proved, we need the following technical definition.

Definition 2.2 Let n be an integer and let $p[n]$ be the smallest prime greater or equal to n. Then, for every integer s, let $\omega_n(s)$ be defined as follows. Let j be the unique integer satisfying $j \equiv s \pmod{p[n]}$ and $1 \leq j \leq p[n]$. Then, $\omega_n(s) = j$, if $1 \leq j \leq n$, and $\omega_n(s) = 1$ otherwise.

Definition 2.3 For every n, the Boolean function f_n is defined for every $x \in \{0,1\}^n$ as $f_n(x) = x_j$, where $j = \omega_n\left(\sum_{i=1}^{n} i\, x_i\right)$.

Using Theorem 2.1, proving a lower bound for the function f_n may be reduced to proving an upper bound to $\nu(f_n, I)$ for some sets I. In Theorem 2.5, we show that $\nu(f_n, I) = 1$ for all sets I of size at most $n - s$ for some $s = o(n)$. For this, we shall use the following theorem originally proved in [4]. A different proof of this theorem may be found in [1] and both proofs in [8].

Theorem 2.4 (Dias da Silva and Hamidoune) *Let p be a prime and let k and h be integers. Moreover, let $h \leq k \leq p$ and let $A \subseteq Z_p$ such that $|A| = k$. Let B be the set of all sums of h distinct elements of A. Then, $|B| \geq \min(p, hk - h^2 + 1)$.*

Theorem 2.5 *For every $I \subseteq \{1, 2, \ldots, n\}$, $|I| \leq n - k(n) - 2$, we have $\nu(f, I) = 1$,where $k(n) = \left\lceil \sqrt{4\, p[n] - 3} \right\rceil$.*

Proof: Throughout this proof, the symbol $\equiv$ means the congruence relation modulo $p[n]$. For simplicity let $k = k(n)$. Moreover, let $I \subseteq \{1, 2, \ldots, n\}$, $|I| \leq n - k - 2$ and let u, v be different partial inputs defined on I. Let $J = \{1, 2, \ldots, n\} \setminus I$ and let

$$\Delta = \sum_{i \in I} i\, v_i - \sum_{i \in I} i\, u_i.$$

We are going to prove $f|_u \neq f|_v$ by finding an extension x of u and an extension y of v such that x and y coincide on the positions in J and $f(x) \neq f(y)$.

Case 1: $\Delta \not\equiv 0$.

We extend u and v to u' and v' by setting some positions not fixed in u and v. First we choose any $j \in J \setminus \{1\}$. Let $l = \omega_n(j + \Delta)$. We have $j \neq l$, since either $l \equiv j + \Delta \not\equiv j$ or $l = 1 \neq j$. Recall that $j \in J$. If also $l \in J$, in u' and v' we set the position j to 0 and the position l to 1. Hence, in this case, we have $u'_j = v'_j \neq u'_l = v'_l$. If $l \in I$, we set the position j of both u' and v' in such a way that $u'_j = v'_j \neq v'_l$.

We have at least k positions that are still not specified in both u' and v'. Let A be a set of k of them; the remaining ones of them we set to zero in both u',v'. Moreover, let $h = \lfloor k/2 \rfloor$. It is easy to verify that $h(k - h) \geq p[n] - 1$. By Theorem 2.4, there is a set $H \subseteq A$ of size h such that

$$\sum_{i \in H} i \equiv j - \sum_{i \in I \cup \{j,l\}} i\, u'_i.$$

Let x extend u' so that $x_i = 1$ for all $i \in H$ and $x_i = 0$ for all $i \in A \setminus H$. Then, we have

$$\sum_{i=1}^{n} i\, x_i \equiv \sum_{i \in H} i + \sum_{i \in I \cup \{j,l\}} i\, u'_i \equiv j.$$

Hence, $f(x) = u'_j$. Let y be a partial input extending v such that the bits with indices in J have the same value in x and y. Then, we have $\omega_n\left(\sum_{i=1}^{n} i\, y_i\right) = \omega_n\left(\sum_{i=1}^{n} i\, x_i + \Delta\right) = l$ and, hence, $f(y) = v'_l \neq f(x)$.

Case 2: $\Delta \equiv 0$.

Let $j \in I$ be such that $u_j \neq v_j$. Let J be the complement of I. Since $|J| \geq k$, Theorem 2.4 may be used as above to prove the existence of an extension x of u such that $\sum_{i=1}^{n} i\, x_i \equiv j$. Let y be a partial input extending v such that the bits with indices in J have the same value in

x and y. According to our assumptions, we have $\sum_{i=1}^{n} i\, y_i \equiv j$. Hence $f(y) = v_j \neq u_j = f(x)$. □

Now, the main result follows from Theorems 2.5 and 2.1, since $p[n] = n + o(n)$.

Theorem 2.6 *There is a sequence of Boolean functions $\{f_n\}_{n=1}^{\infty}$ that is in P and such that f_n is a function of n variables and for every n large enough, every 1–b. p. computing f_n has size at least $2^{n-3\sqrt{n}}$.*

In the present paper we use the weighted sum $\sum_{i=1}^{n} i\, x_i$ to define the function f_n. In an earlier version of the paper a weighted sum $\sum_{i=1}^{n} w_i\, x_i$ was used to prove a lower bound of the size 2^{n-s} where $s = n^{2/3} \ln^{1/3} n$ without an application of Theorem 2.4.

For completness let us mention the following facts without proof. The sequence $\{f_n\}$ from our theorem is computable on $(1,+1)$–b. p.'s within the size $O(n^2)$. Hence we have a separation between 1–b. p.'s and $(1,+1)$-b.p.'s. Similarly, on nondeterministic 1–b. p.'s $O(n^3)$ vertices are sufficient.

Theorem 2.6 may be compared with the maximal complexity of 1–b. p.'s. Every function has a 1–b. p. of size $2^{n-\log n+O(1)}$ and there is a nonconstructive proof that some functions require size $2^{n-\log n-O(1)}$, see [16].

3 A possible generalization

The function f_n from the previous section is defined as x_j, where j is computed from a weighted sum of the bits in the input vector x. The basic property of this weighted sum used in the proof of Theorem 2.5 was that if some setting leaves at least $3\sqrt{n}$ free input bits, it is still possible to set these bits in such a way that the weighted sum belongs to any given residue class modulo $p[n]$. In [15], Szemeredi proved that for any weighted sum of the input bits with any integer coefficients, there is a setting that leaves $\Omega(\sqrt{n})$ free bits that are not sufficient to reach any residue class modulo $p[n]$. Hence, in order to get better lower bounds, it is necessary to use a different approach.

In this section, we shall investigate the possibility to replace the weighted sum of input bits by a more general function. More exactly, we shall investigate the complexity of functions in the form $x_{\phi(x)}$ on 1–b. p.'s, where ϕ is a function of the type $\phi : \{0,1\}^n \to \{1,2,\ldots,n\}$.

If I is any subset of the indices of the input bits, u is a partial input from $\mathcal{B}(I)$ and g is a function on the domain $\{0,1\}^n$ and any range, then let $g|_u$ mean the function of $n - |I|$ Boolean variables obtained from g by setting the bits specified in u according to u.

Definition 3.1 Let n, s and q be integers and let $\phi : \{0,1\}^n \to \{1,2,\ldots,n\}$ be a function. Then, we say that ϕ is (s,n,q)-complete, if for every $I \subseteq \{1,2,\ldots,n\}$, $|I| = n - s$, we have:
(i) For every $u \in \mathcal{B}(I)$, the restricted function $\phi|_u$ has all the values from $\{1,2,\ldots,n\}$ in its range.
(ii) There is at most q different partial inputs u from $\mathcal{B}(I)$ giving different subfunctions $\phi|_u$.

The condition (i) is a straightforward generalization of the basic property of the weighted sum used in the previous section. It appears, however, that this condition is not sufficient to prove a good lower bound in a way similar to the application of the weighted sum. The reason is that in the proof of Theorem 2.5, we essentially use the fact, that we work with a weighted sum, i.e. with a linear combination of the input bits. In order to work with a general function ϕ, we include condition (ii). Using this condition, the proof is slightly different from the use of the weighted sum, but it yields almost the same lower bound.

Theorem 3.2 *Let ϕ be (s,n,q)–complete. Then, every 1–b. p. computing $x_{\phi(x)}$ has size at least $2^{n-s}/q$.*

Proof: Denote by f the function $x_{\phi(x)}$. In order to prove the theorem, we shall prove that for any I satisfying $|I| = n - s$ we have $\nu(f, I) \leq q$. Then, the required lower bound follows from Theorem 2.1.

Let I be such that $|I| = n - s$. Assume for a moment that $\nu(f, I) > q$. Then, there are partial inputs $u_1, u_2, \ldots, u_{q+1}$ defined on I such that $f|_{u_i}$ is the same subfunction for all $i = 1, 2, \ldots, q+1$. By the assumption (ii) on ϕ, there are some indices i and j such that $\phi|_{u_i} = \phi|_{u_j}$. Let $k \in I$ be an index of a bit, where u_i and u_j differ. By the assumption (i) on ϕ, there is a partial input x defined on the complement of I such that $\phi([u_i, x]) = \phi([u_j, x]) = k$. Then, $f([u_i, x]) \neq f([u_j, x])$, a contradiction with the choice of $u_1, u_2, \ldots, u_{q+1}$. □

Note that $f_n = x_{\phi(x)}$, where $\phi(x) = \omega_n\left(\sum_{i=1}^{n} i\, x_i\right)$. It is easy to see that this function is $(\lceil 3\sqrt{n}\,\rceil, n, n)$–complete. This together with Theorem 3.2 yields a lower bound that is worse than the lower bound of Theorem 2.6 only by a factor of n. It is an open problem if an explicitly defined (s,n,q)–complete function ϕ may be constructed for some $s = o(\sqrt{n})$ and $q = 2^{o(\sqrt{n})}$.

In the rest of this section, we exhibit a nonconstructive proof of the fact that for every n an (s,n,q)–complete function ϕ exists with $s = O(\log n)$ and $q = n^{O(1)}$.

Lemma 3.3 *Let A be a $t \times n$ matrix over $GF(2)$ such that every $t \times s$ submatrix has rank at least r. Let $\psi : \{0,1\}^t \to \{1, 2, \ldots, n\}$ be such that on every affine subset of $\{0,1\}^t$ of dimension at least r, it reaches all the values from $\{1, 2, \ldots, n\}$. Then, $\phi(x) = \psi(Ax)$ is $(s, n, 2^t)$–complete.*

Proof: To prove the property (ii) of Definition 3.1, note that the contribution to Ax of setting any subset of the input bits to some constants is expressible as a vector of length t over $GF(2)$. Hence, at most $q = 2^t$ different subfunctions $\phi|_u$ are possible for $u \in \mathcal{B}(I)$ for any fixed I.

In order to prove property (i), note that if $|I| = n - s$ and $u \in \mathcal{B}(I)$, then the set of vectors $A \times [u, x]$ of length t for all settings $x \in \mathcal{B}(J)$, where $J = \{1, 2, \ldots, n\} \setminus I$, is an affine subset of $\{0,1\}^t$ of dimension equal to the rank of the $t \times s$ submatrix of A formed by columns with indices in J. By our assumption on A, this dimension is at least r. Hence, by the assumption on ψ, every value from $\{1, 2, \ldots, n\}$ is equal to $\psi(A \times [u, x])$ for an appropriate $x \in \mathcal{B}(J)$. □

Theorem 3.4 *For every n large enough, there is an (s,n,q)–complete mapping $\phi : \{0,1\}^n \to \{1, 2, \ldots, n\}$ for some $s = O(\log n)$ and $q = n^{O(1)}$.*

Proof: Let $\varepsilon > 0$ and let s, t and r be given by the formulas:

$$s = \lceil (2 + 3\,\varepsilon) \log n \rceil$$

$$t = \lceil (3 + 3\,\varepsilon) \log n \rceil$$

$$r = \lceil (1 + 2\,\varepsilon) \log n \rceil.$$

We are going to prove that a random $t \times n$ matrix A and a random mapping $\psi : \{0,1\}^t \to \{1, 2, \ldots, n\}$ satisfy the assumptions of Lemma 3.3 with positive probability. This implies that a matrix A and a mapping ψ satisfying the assumptions of Lemma 3.3 with the given

parameters s, t and r exist. For such A and ψ, the mapping $\psi(Ax)$ is (s,n,q)–complete with $q = 2^t$ by Lemma 3.3. Since we have $s = O(\log n)$ and $t = O(\log n)$, this will imply the theorem.

Let A be chosen at random. Choose some $t \times s$ submatrix of the matrix A. If this submatrix has rank at most r, then there is a set of $s - r$ of its columns that are in the linear span of the remaining r columns. For a fixed choice of these $s - r$ columns, this happens with probability at most $(2^r/2^t)^{s-r}$. Hence, the total probability of the event that A contains a $t \times s$ submatrix of rank at most r is at most

$$\binom{n}{s}\binom{s}{r}2^{-(t-r)(s-r)}.$$

The logarithm of this is at most

$$s\log n + r\log s - (t-r)(s-r) = -\varepsilon^2\log^2 n + O(\log n \log\log n) \to -\infty.$$

Hence, the probability that A does not satisfy the requirements of Lemma 3.3 converge to zero.

Let ψ be chosen at random. Every affine subset of $\{0,1\}^t$ of dimension r is determined by some of its elements and a linear subspace of dimension r of $\{0,1\}^t$. The subspace is determined by some set of its generators. Hence, the number of affine subsets of dimension r is at most $2^{(r+1)t}$. (This estimate will be sufficient for our purpose. A better calculation yields an upper bound $O(2^{(r+1)(t-r)})$.) The probability that a random mapping misses some of the values from $\{1,2,\ldots,n\}$ on some subset of size 2^r of its domain is at most $n(1-1/n)^{2^r}$. Hence, the total probability that ψ does not satisfy our requirements is at most

$$2^{(r+1)t}n\left(1-\frac{1}{n}\right)^{2^r} \le 2^{O(\log^2 n)}\mathrm{e}^{-2^r/n} \le 2^{O(\log^2 n)}\mathrm{e}^{-n^{2\varepsilon}}.$$

Since this converges to zero, the theorem is proved. □

Note that Theorem 3.4 implies existence of ϕ, for which Theorem 3.2 yields a lower bound $2^{n-O(\log n)}$.

Acknowledgement The authors are grateful to Endre Szemeredi for directing their attention to [1], [4] and [8].

References

[1] N. Alon, M. B. Nathanson and I. Z. Ruzsa, The polynomial method and restricted sums of congruence classes, *J. Number Theorey*, to appear.

[2] L. Babai, P. Hajnal, E. Szemeredi and G. Turan, A lower bound for read-once-only branching programs, *Journal of Computer and Systems Sciences*, vol. 35 (1987), 153–162.

[3] A. Borodin, A.Razborov and R. Smolensky, On Lower Bounds for Read-k-times Branching Programs, *Computational Complexity* 3 (1993) 1 – 18.

[4] J. A. Dias da Silva and Y. O. Hamidoune, Cyclic spaces for Grassmann derivatives and additive theory, *Bull. London Math. Soc.*, 26 (1994), 140–146.

[5] P. E. Dunne, Lower bounds on the complexity of one–time–only branching programs, In *Proceedings of the FCT, Lecture Notes in Computer Science*, 199 (1985), 90–99.

[6] S. Jukna, A Note on Read-k-times Branching Programs, *RAIRO Theoretical Informatics and Applications*, vol. 29, Nr. 1 (1995), pp. 75–83.

[7] S. Jukna, A. A. Razborov, Neither Reading Few Bits Twice nor Reading Illegally Helps Much, TR96-037, ECCC, Trier.

[8] M. B. Nathanson, *Additive Number Theory: 2. Inverse Theorems and the Geometry of Sumsets.* Graduate Texts in Mathematics, Springer-Verlag, New York, 1995.

[9] E. A. Okolnishkova, Lower bounds for branching programs computing characteristic functions of binary codes (in Russian), *Metody diskretnogo Analiza,* 51 (1991), 61–83.

[10] S. J. Ponzio, A lower bound for integer multiplication with read-once branching programs, *Proceedings of 27's Annual ACM Symposium on the Theory of Computing*, Las Vegas, 1995, pp. 130–139.

[11] P. Savický, S. Žák, A Lower Bound on Branching Programs Reading Some Bits Twice, to appear in TCS.

[12] D. Sieling, New Lower Bounds and Hierarchy Results for Restricted Branching Programs, TR 494, 1993, Univ. Dortmund, to appear in *J. of Computer and System Sciences.*

[13] D. Sieling and I. Wegener, New Lower bounds and hierarchy results for Restricted Branching Programs, in *Proc. of Workshop on Graph-Theoretic Concepts in Computer Science WG'94*, Lecture Notes in Computer Science Vol. 903 (Springer,Berlin, 1994) 359 – 370.

[14] J. Simon, M. Szegedy, A New Lower Bound Theorem for Read Only Once Branching Programs and its Applications, *Advances in Computational Complexity Theory* (J. Cai, editor), DIMACS Series, Vol. 13, AMS (1993) pp. 183–193.

[15] E. Szemeredi, personal communication.

[16] I. Wegener, *The complexity of Boolean functions*, Wiley-Teubner Series in Computer Science, 1987.

[17] I. Wegener, On the Complexity of Branching Programs and Decision Trees for Clique Functions, *JACM* 35 (1988) 461 – 471.

[18] I. Wegener, Efficient data structures for the Boolean functions, *Discrete Mathematics* 136 (1994) 347 – 372.

[19] S. Žák, An Exponential Lower Bound for One-time-only Branching Programs, in *Proc. MFCS'84*, Lecture Notes in Computer Science Vol. 176 (Springer, Berlin, 1984) 562 – 566.

[20] S. Žák, A superpolynomial lower bound for $(1, +k(n))$- branching programs, in *Proc. MFCS'95*, Lecture Notes in Computer Science Vol. 969 (Springer, Berlin, 1995) 319 – 325.