A Hierarchy for (1,+k)-branching programs with respect to k

Savický, Petr
1996

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

A hierarchy for $(1, +k)$–branching programs with respect to k

P. Savický, S. Žák

Technical report No. 672

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic
phone: (+422) 66414244 fax: (+422) 8585789
e-mail: {stan,savicky}@uivt.cas.cz

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

A hierarchy for $(1,+k)$–branching programs with respect to k

P. Savický, S. Žák [1]

Technical report No. 672

Abstract

Branching programs (b. p.'s) or decision diagrams are a general graph-based model of sequential computation. The 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 so called k-b. p.'s, where each computation reads each input bit at most k times. Although, for more restricted syntactic k-b.p.'s, exponential lower bounds are proven and there is a series of exponential lower bounds for 1-b. p.'s, this is not true for general (nonsyntactic) k-b.p.'s, even for $k = 2$. Therefore, so called $(1,+k)$-b. p.'s are investigated.

For some explicit functions, exponential lower bounds for $(1,+k)$-b. p.'s are known. Investigating the syntactic $(1,+k)$-b. p.'s, Sieling has found functions $f_{n,k}$ which are polynomially easy for syntactic $(1,+k)$-b. p.'s, but exponentially hard for syntactic $(1,+(k-1))$-b. p.'s. In the present paper, a similar hierarchy with respect to k is proven for general (nonsyntactic) $(1,+k)$-b. p.'s.

Keywords

branching programs, lower bounds

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

1 Introduction

A branching program (b. p.) is a computation model for representing the Boolean functions. The input of a branching program is a vector consisting of the values of n Boolean variables. 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 input variable 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. 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 variable assigned to the node is tested and the out-going edge labeled by the actual value of the variable 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.

The 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. Hence, 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. An important restriction are so called read-once branching programs (1–b. p.'s), where the restriction is such that during each computation on any input each variable is tested at most once.

The first exponential lower bounds for 1-b. p.'s were [16] and [17]. These results were improved in [5]. These bounds are of magnitude $2^{\Omega(\sqrt{n})}$. The first lower bound of magnitude $2^{\Omega(n)}$, where n is the input size was $2^{n/c}$ lower bound for a large c for the function "parity of the number of triangles in a graph", see [2]. The constant c in this lower bound was improved, see [15], to a lower bound of magnitude $2^{n/2000}$ for the same function. For a different function, a lower bound $2^{n-o(n)}$ was proved in [12]. In [9], a lower bound $2^{c\sqrt{n}}$ is proved for multiplication.

Several generalizations of 1-b. p.'s are investigated. Recently, the most powerful among them are so called k-b. p.'s, where each computation is allowed to test each variable at most k times. Since no superpolynomial lower bounds even for 2-b. p.'s are known, even more restricted b. p.'s are investigated. Namely, so called $(1,+k)$-b. p.'s, where for every input, there are at most k variables that are tested during the computation more than once. If, moreover, the variables with repeated tests may be read at most two times, we obtain a class that contain 1-b. p.'s and is contained in 2-b. p.'s. For $(1,+k)$-b. p.'s, an exponential lower bound for k up to n^{α} for some fixed positive α may be found in [7], [11] and [18]. The results of [7] and [11] hold even for α arbitrarily close to 1. A superpolynomial lower bound for $k = o(n/\log n)$ is proved in [7].

The two restrictions mentioned above, namely k-b. p.'s and $(1,+k)$-b. p.'s can be made even stronger, if the restriction of repeated tests is applied not only to valid computation paths, but to every possible path from the source to a sink in the b. p., including inconsistent paths. In this way, we obtain so called syntactic k-b. p.'s and syntactic $(1,+k)$-b. p.'s. Note that each 1–b. p. is a syntactic 1–b. p., while for k–b. p.'s, $k \geq 2$ this is not true in general.

For syntactic k-b. p.'s, exponetial lower bounds are known, see [3], [6], [8], for some $k = \Omega(\log n)$. The result of [3] and [6] hold even for nondeterministic k-b. p.'s. For syntactic $(1, +k)$-b. p.'s, a strict hierarchy according to k was proved in [13] and [14] for k at most roughly $n^{1/3}$.

In the present paper, we generalize the last mentioned result in such a way that it holds also for (nonsyntactic) b. p.'s for $k \leq 1/2 \cdot n^{1/8} \log^{-1/4} n$. Namely, we present functions $f_{n,k}$ which are polynomially easy for $(1, +k)$-b. p.'s, but exponentially hard for $(1, +(k-1))$-b. p.'s.

In order to prove the hierarchy result, an exponential lower bound for $(1, +k)$-b. p.'s is proved. In comparison to [7], [11] and [18] the exponential lower bound is reached in a smaller range of k. On the other hand, while all the three mentioned results use in fact the same method, the present paper is based on a different method. The method of [7], [11] and [18] may be applied only to functions f satisfying the following requirement or its dual. If $f(x) = f(y) = 1$, then either $x = y$ or the Hamming distance of x and y is at least n^{ε} for some positive ε. The function used for the lower bound in the present paper does not have this property.

The structure of the paper is as follows. In Section 2, a function $f_{n,k}$ of n variables is defined and some its properties are proved. In Section 3 an exponential lower bound for $f_{n,k}$ in $(1, +(k-1))$–b. p.'s is proved using two theorems proved later in Sections 5 and 6. In Section 4, we present a polynomial size $(1, +k)$-b. p. for $f_{n,k}$ and summarize the main result.

2 The function and its basic properties

In this section, we define the function $f_{n,k}$ and prove some properties of the function. Informally, the function is defined as follows. The n variables are divided into k blocks of length m. For every $j = 1, 2, \ldots, k$, a weighted sum of the bits of block j determines an index i_j of some of the input bits. Then, the value of the function is the parity of the bits determined by i_j for $j = 1, 2, \ldots, k$. The exact definition of $f_{n,k}$ requires some technical notation.

For every natural number n, let $p(n)$ be the smallest prime greater than n. Consider the set $\{1, 2, \ldots, n\}$ as a subset of $Z_{p(n)}$, the field of the residue classes modulo $p(n)$. Then, for every $t \in Z_{p(n)}$, let $\omega(t) = t$, if $t \in \{1, 2, \ldots, n\}$ and $\omega(t) = 1$ otherwise.

Definition 2.1 For every $t = (t_1, t_2, \ldots, t_k) \in \{1, 2, \ldots, n\}^k$ and every $x \in \{0,1\}^n$, let $Par(x, t) = x_{t_1} \oplus x_{t_2} \oplus \ldots \oplus x_{t_k}$.

Definition 2.2 Let k divide n and let $m = n/k$. Then, let $\psi_{n,k} : \{0,1\}^n \to \{1, 2, \ldots, n\}^k$ be defined as follows. For every x let $\psi_{n,k}(x) =_{\text{def}} (t_1, t_2, \ldots, t_k)$, where for every $j = 1, 2, \ldots, k$,

$$t_j = \omega\left(\sum_{i=1}^{m} i\, x_{(j-1)m+i}\right),$$

where the sum is evaluated in $Z_{p(n)}$. Moreover, let $f_{n,k}(x) =_{\text{def}} Par(x, \psi_{n,k}(x))$.

In order to prove some required properties of $\psi_{n,k}$, we shall use the following theorem originally proved in [4]. A different proof of this theorem may be found in [1].

Theorem 2.3 (Dias da Silva and Hamidoune) *Let p be a prime and let h_1 and h_2 be integers. Moreover, let $h_2 \leq h_1 \leq p$ and let $A \subseteq Z_p$ such that $|A| = h_1$. Let A' be the set of all sums of h_2 distinct elements of A. Then, $|A'| \geq \min(p, h_2(h_1 - h_2) + 1)$.*

Corollary 2.4 *Let $\varepsilon > 0$ be fixed. Then, for every n large enough, the following is true. If $A \subseteq Z_{p(n)}$ and $|A| \geq (2+\varepsilon)\sqrt{n}$, then, for every $t \in Z_{p(n)}$, there is a subset $B \subseteq A$ such that the sum of the elements of B is equal to t.*

Proof: Let $h_2 = \lfloor (1+\varepsilon/2)\sqrt{n} \rfloor$ and $h_1 = 2h_2$. Choose any subset C of A of size h_1. By Theorem 2.3, there is at least $\min(p(n), h_2^2 + 1)$ different sums of h_2 distinct elements of C. We have $h_2^2 + 1 \geq (1+\varepsilon)n - O(\sqrt{n})$. Since $p(n) = n + o(n)$, see [10], we have $\min(p(n), h_2^2 + 1) = p(n)$. Hence, for every $t \in Z_{p(n)}$, there is a set $B \subset A$ of h_2 elements adding up to t. $\Box$

Lemma 2.5 *For any fixed $\varepsilon > 0$, for every n large enough and for every k as above, the following is true. If r is an integer such that $n > r \geq k((2+\varepsilon)\sqrt{n} + 1)$, and if at most $n - r$ of the variables of the function $Par(x, \psi_{n,k}(x))$ are set to some constants, the restricted function is still not a constant function.*

Proof: For simplicity, let $s = \lfloor (2+\varepsilon)\sqrt{n} \rfloor$. There are at least $r \geq k(s+1)$ free variables distributed among the k blocks. Let t be the number of blocks with at most s free variables. Set the free variables in these t blocks in an arbitrary way, say, to zeros. Now, these t blocks contain only fixed variables and it is possible to evaluate the entries of $\psi_{n,k}(x)$ determined by these blocks. There are at most t different input variables the index of which is equal to some of these indices. Let us call these variables marked.

The remaining blocks contain at least $(k-t)(s+1)+t$ free variables and, hence, at least $(k-t)(s+1)$ free variables that are not marked. It follows that there is a block with at least $s+1$ free variables that are not marked. Choose $s+1$ of these variables, call them critical variables, and set the other free variables in the same block in an arbitrary way, say to zeros.

There are still $k-t-1$ blocks with some free variables different from the block with the critical variables. Each of these blocks contain at least $s+1$ free variables, marked or not marked. Set all free variables in each of these blocks in such a way that the index determined by each of these $k-t-1$ blocks belongs to the block itself. This is possible according to Corollary 2.4.

Now, the only remaining free variables are the critical variables and all these variables are in one block. The indices determined by the other blocks are fixed to values different from the indices of critical variables. The function obtained by this setting of variables is not a constant function, since it is possible to set one of the critical variables to 0 or 1 and the remaining ones in such a way that the index determined by the block with critical variables is just the index of the first chosen critical variable (Corollary 2.4). Hence, the value of the function may be both 0 and 1. $\Box$

Lemma 2.6 *If at most $n/k - 3\sqrt{n}$ of the variables of the function $\psi_{n,k}$ are set to some constants, the restricted function still satisfies the following. For every choice of $1 \leq i_1, i_2, \ldots, i_k \leq n$, there is a setting of the free variables such that the value of $\psi_{n,k}$ is equal to $(i_1, i_2, \ldots, i_k)$.*

Proof: In each blok, at least $3\sqrt{n}$ variables are free. Using Corollary 2.4, each entry of $\psi_{n,k}(x)$ may be set to any value from $\{1, 2, \ldots, n\}$ independently of the other entries. □

3 The lower bound

In this section, we prove an exponential lower bound for $f_{n,k}$ in $(1, +(k-1))$-b. p.'s, if k is not too large. There are several possibilities how to bound the number of repeated tests in a path. We use the following definition, i.e., we count only the number of different variables involved in the repeated tests, not the number of these tests.

Definition 3.1 *Let P be a b. p. For every input x, let $R(x)$ be the set of indices of input bits that are read more than once during the computation for x. The b. p. P is called a $(1, +k)$-b. p., if for every x, $|R(x)| \leq k$.*

For a path α, let $\pi(\alpha)$ be the set of variables tested in α.

Definition 3.2 Let S be some set of paths in a b. p. going from a node u to a node v. Then the number

$$\left| \bigcup_{\alpha \in S} \pi(\alpha) \right| - \min_{\alpha \in S} |\pi(\alpha)|$$

will be called the *fluctuation* of S. If both occurrences of $\pi(\alpha)$ in the expression above are replaced by $\pi(\alpha) \cap I$, where I is a set of variables, we call the resulting number the fluctuation of S relative to I.

We say that an edge (u, v) is a *test* of a variable x_i, if u is of degree 2 and x_i is the label of u.

Definition 3.3 We say that a branching program P is (p, r)-*well-behaved*, if it satisfies the following three conditions:
(i) Every path from the source of P to a sink contains at least p tests.
(ii) The first p tests on any path starting in the source of P test p different variables.
(iii) If w is any node of P and S is the set of all paths with p tests leading from the source to w, then the fluctuation of S is less than r.

One of the key steps of the proof is the following theorem. Its proof may be found in Section 5.

Theorem 3.4 *Let n, k, p, r be integers such that $r < n$, $2kp \leq n - r$. Let f be a Boolean function of n variables such that any setting of at most $n - r$ variables to constants still leads to a nonconstant function. Let P be a $(1, +(k-1))$–b. p. computing f. Then, there is a subprogram P' of P arising from P by setting at most $(2k-1)p$ variables to constants that is (p, r)-well-behaved.*

To prove the lower bound, we shall combine Theorem 3.4 with the following Theorem 3.5. This theorem is implicitly used already in [13]. For convenience of the reader, we present a complete proof in Section 6.

If v_j is a vector, let $v_{j,i}$ be its i–th coordinate.

Theorem 3.5 *Let k, p and m be integers and let k divide p. For $j = 1, 2, \ldots, m$, let $v_j \in \{0,1\}^I$, where I is some index-set of size p. Assume that for every k–tuple $i_1, i_2, \ldots, i_k \in I$, there is a function $\phi : \{0,1\}^I \to \{0,1\}$ such that*
(1) ϕ is computable by a decision tree of depth at most $k-1$.
(2) For every $j = 1, 2, \ldots, m$ we have

$$\phi(v_j) = v_{j,i_1} \oplus v_{j,i_2} \oplus \ldots \oplus v_{j,i_k}.$$

Then, we have $m \leq 2^{p(1-1/k)}$.

Now, we can state and prove the lower bound result.

Theorem 3.6 *Let n, k be integers, let k divides n and let $k \leq \sqrt{n}/3$. Then, every $(1, +(k-1))$-b. p. computing $f_{n,k}$ has size at least*

$$2^{\left(\frac{n}{2k^3} - \frac{3\sqrt{n}}{2k^2} - 3k\sqrt{n}\log n - 1\right)}.$$

Proof: First, let us introduce an auxiliary notation. For partial inputs $u_1, u_2, \ldots, u_s$ specifying disjoint sets of bits, let $[u_1, u_2, \ldots, u_s]$ denote the (partial) input specifying all the bits specified in some of u_j in the same way as in corresponding u_j.

Let $r = k(\lfloor 3\sqrt{n} \rfloor - 1)$, $q = n/k - 3\sqrt{n}$ and $p = \lfloor q/(2k^2) \rfloor k$. We have $2kp \leq q \leq n - r$. By Lemma 2.5, setting of at most $n - r$ variables in $Par(x, \psi_{n,k}(x)) = f_{n,k}$ leads to a nonconstant function. Hence, the function $Par(x, \psi_{n,k}(x))$ satisfies the assumption of Theorem 3.4 for our choice of k, p and r. Let P be a $(1, +(k-1))$–b. p. of size c computing $f_{n,k}$. Consider the subprogram P' of P guaranteed by Theorem 3.4. Let u be the partial input with at most $(2k-1)p$ fixed variables which yields P' and let w_1 be the source of P'. We have that P' is a $(1, +(k-1))$–b. p. of size at most c computing the restriction of $Par(x, \psi_{n,k}(x))$ according to u.

Let w_2 be the node of P' such that the number of paths starting at w_1, ending in w_2 and containing p tests is maximal. There is at least $2^p/c$ of such paths. Call the set of these paths $\mathcal{P}_1$. Each path tests some set of variables. Since P' is (p, r)-well-behaved, the fluctuation of $\mathcal{P}_1$ is less than r and hence, there are at most $\binom{p+r}{r}$ of different sets of variables tested along individual paths from $\mathcal{P}_1$. Let $\mathcal{P}_2$ be some of the largest subsets of $\mathcal{P}_1$ of paths testing exactly the same set of variables. Then, we have

$$|\mathcal{P}_2| \geq \frac{2^p}{c\binom{p+r}{r}}.$$

Each path in $\mathcal{P}_2$ together with u determines a partial input. For every partial input, it is possible to evaluate its contribution to the k entries of the value of $\psi_{n,k}$. By this, we mean the sums from Definition 2.2 restricted to bits with the value fixed by u and the given path from $\mathcal{P}_2$. The number of possible contributions is at most n^k. Hence, there is a subset $\mathcal{P}_3$ of $\mathcal{P}_2$ of paths with the same contributions and such that its size $m =_{\text{def}} |\mathcal{P}_3|$ satisfies

$$m \geq \frac{2^p}{c\binom{p+r}{r} n^k}. \tag{3.1}$$

Let $v_1, v_2, \ldots, v_m$ be the list of elements of $\mathcal{P}_3$ and let I be the set of indices of variables set to a constant by inputs v_j. By construction of $\mathcal{P}_2$, $|I| = p$. We are going to verify that the inputs $v_1, v_2, \ldots, v_m$ satisfy the assumption of Theorem 3.5.

Let us fix some $i_1, i_2, \ldots, i_k \in I$. Let x be a partial input such that $[u, v_1, x]$ is a total input satisfying $\psi_{n,k}([u, v_1, x]) = (i_1, i_2, \ldots, i_k)$. Such an x exists, since the number of bits fixed by $[u, v_1]$ is not larger than $2kp \leq q \leq n/k - 3\sqrt{n}$ and therefore we may apply Lemma 2.6.

Since all the partial inputs $v_1, v_2, \ldots, v_m$ have the same contributions to the sums in the Definition 2.2, $\psi_{n,k}([u, v_j, x]) = (i_1, i_2, \ldots, i_k)$ for all $j = 1, 2, \ldots, m$.

Consider the restriction P'' of P' according to the values of input bits from the input x. The only free input bits of P'' are the bits from I. For every v_j, the computation of P'' computes $f_{n,k}([u, v_j, x]) = Par([u, v_j, x], (i_1, i_2, \ldots, i_k)) = v_{j,i_1} \oplus v_{j,i_2} \oplus \ldots \oplus v_{j,i_k}$. Moreover, for every v_j, the computation of P'' reads all the bits from I, then it reaches the node w_2 and in the rest of the computation, at most $k-1$ variables with indices in I are read. (Since P'' is also a $(1, +(k-1))$-b. p.)

Consider the subprogram of P'' starting in w_2 and let P''' be the decision tree obtained from this subprogram as follows. First, we expand the subprogram starting at w_2 into a tree. In the second step, we delete all edges of the tree that are not visited by any computation starting from w_2 for some of v_j. After this, some of the nodes of the tree might have out-degree 1. In the last step, every such node is deleted and the edge leading to it is redirect to the single successor of the considered node.

Every leaf of P''' is reached by a computation for some v_j, otherwise some of the edges of the path leading to the leaf would have been deleted. Hence, each path of P''' tests at most $k-1$ variables, since it tests a subset of the set of variables read by some computation of P'' after the node w_2 for some of v_j.

Let ϕ be the function computed by P'''. Clearly, ϕ satisfies the assumption (1) of Theorem 3.5.

By construction of P''', P'' and P''' are equivalent on inputs v_j. Thus, for each $j = 1, 2 \ldots, m$, we have $\phi(v_j) = f_{n,k}([u, v_j, x]) = v_{j,i_1} \oplus v_{j,i_2} \oplus \ldots \oplus v_{j,i_k}$. Hence, ϕ satisfies also the assumption (2) of Theorem 3.5.

These arguments work for every k–tuple of indices from I. Hence, Theorem 3.5 implies $m \leq 2^{p(1-1/k)}$.

Together with (3.1), this implies

$$c \geq \frac{2^{p/k}}{\binom{p+r}{r} n^k}.$$

Since $\binom{p+r}{r} \leq n^r$, we have

$$c \geq 2^{p/k - r\log n - k\log n}. \tag{3.2}$$

The theorem now follows by substitution of the chosen values of p, r and k into the last estimate. □

4 The hierarchy

We shall prove an upper bound for the function $f_{n,k}$ on $(1, +k)$-b. p.'s. Together with the lower bound from the previous section, it gives that $(1, +k)$-b. p.'s are more

powerful than $(1,+(k-1))$-b. p.'s.

Theorem 4.1 *Let $k = k(n) \leq 1/2 \cdot n^{1/8} \log^{-1/4} n$. Then, for every n large enough, we have (i) There is a $(1,+k)$-b. p. computing $f_{n,k}$ of size $O(n^2)$.*
(ii) Every $(1,+(k-1))$-b. p. computing $f_{n,k}$ has size at least $2^{\Omega(n/k^3)}$.

Proof: Let us start with (i). We shall construct a $(1,+k)$-b.p. P computing $f_{n,k}$. Consider the input bits in the input x divided into k groups in the same way as in the definition of $\psi_{n,k}$. Let $\psi_{n,k}(x) = (i_1, i_2, \ldots, i_k)$. In order to describe P, we shall describe for every $j = 1, 2, \ldots, k$ a b. p. P_j computing $x_{i_1} \oplus x_{i_2} \oplus \ldots \oplus x_{i_j}$. Then, P is P_k.

Let $x_1, x_2, \ldots, x_m$ be the bits in the first group. The b. p. P_1 is leveled and it reads the bits in the first group in the natural ordering. For simplicity, assume that each level consists of $p(n)$ vertices corresponding to the residue classes mod $p(n)$. The computation starts in level 0 in the node corresponding to 0. After reading x_j, the computation reaches the j-th level in the node corresponding to the residue class $x_1 + 2x_2 + 3x_3 + \ldots + jx_j \bmod p(n)$. For each $j = 0, 1, \ldots, m-1$ and each node w at level j, this determines the two nodes at level $j+1$, where the edges from w lead to. Consider the node corresponding to $t \in Z_{p(n)}$ at level m. In this node, the variable $\omega(t)$ is tested and its value is the output of P_1.

The b. p. just described computes x_{i_1}, since the computation reaches the m-th level in the node corresponding to $x_1 + 2x_2 + 3x_3 + \ldots + mx_m \bmod p(n)$ and by definition of $\psi_{n,k}$, we have $i_1 = \omega(x_1 + 2x_2 + 3x_3 + \ldots + mx_m)$.

Now, assume, P_j is constructed. In order to construct P_{j+1}, append to each of the two sinks of P_j a b. p., computing x_{i_j} in a way similar to the computation of x_{i_1} in P_1. We obtain a b. p. with four sinks corresponding to the four possible values of $x_{i_1} \oplus x_{i_2} \oplus \ldots \oplus x_{i_j}$ and $x_{i_{j+1}}$. Now, P_{j+1} is obtained by joining the sinks with the same value of $x_{i_1} \oplus x_{i_2} \oplus \ldots \oplus x_{i_{j+1}}$.

Note that, P_1 has size at most $p(n)n/k$. Moreover, for each $j = 2, 3, \ldots, k$, the b. p. P_j contains at most $2p(n)n/k$ additional nodes w.r.t. P_{j-1}. Hence, P_k is of size at most $2p(n)n = O(n^2)$.

In order to prove (ii), note that, if $k = k(n) \leq 1/2 \cdot n^{1/8} \log^{-1/4} n$, then

$$\frac{1}{2} \cdot \frac{n}{2k^3} \geq \frac{3\sqrt{n}}{2k^2} + 3k\sqrt{n} \log n + 1 \,.$$

Using this, Theorem 3.6 implies (ii). □

5 Proof of Theorem 3.3

Let us start the proof by the following. Let an edge (u, v) be a test of a variable x_i. The Boolean value labeling this edge is called the value required by this test. Two tests of the same variable are consistent, if they require the same value.

Now, we shall construct a sequence $v_0, v_1, \ldots, v_t$ of nodes of P, where v_0 is the source and v_t is some of the sinks and a sequence $T_1, T_2, \ldots, T_t$, where T_i is a set of some paths from v_{i-1} to v_i. We shall construct these two sequences by a process starting with v_0 beeing the source of P and with an empty sequence of sets. The process will be

described in steps. In step j, we start with some sequence $v_0, v_1, \ldots, v_{j-1}$ of nodes and a sequence of sets $T_1, T_2, \ldots, T_{j-1}$, we add a new node v_j, a new set T_j and possibly modify the sets T_i for $i = 1, \ldots, j-1$. The process stops, when v_j becomes a sink of P and we set $t = j$. In each step of the process, to each of the sets T_i a *type* A, B or C is assigned. The type is assigned when the set is created and it may be modified, if the set is changed at some later step.

Definition 5.1 Let $T_1, T_2, \ldots, T_j$ be the sets constructed at some step of the process. Let t_1, t_2 be tests of the same variable contained in $T_1, T_2, \ldots, T_j$. Then, we say that the test t_1 *preceeds* the test t_2, if either for some i both t_1 and t_2 are in some path $\alpha \in T_i$ and t_1 preceeds t_2 in α or t_1 is contained in T_{i_1} and t_2 in T_{i_2} and $i_1 < i_2$. Moreover, we say that the *rank* of a test t is h, if h is the maximum integer, for which there are tests $t_1, t_2, \ldots, t_h$ of the same variable, such that $t = t_h$ and for all $i = 2, 3, \ldots, h$, t_{i-1} preceeds t_i. A test is called a *repeated* test, if its rank is at least 2.

In each step of the process, we require that the sequence $T_1, T_2, \ldots, T_j$ is consistent. By this, we mean the following. If $\alpha_i \in T_i$ is any choice of one path from each of the sets, the concatenation of α_i is a consistent path in P. We will require even stronger structural property. Our requirement is as follows.

Requirement 1 Let x_i be any variable contained in a repeated test in $T_1, T_2, \ldots, T_j$. Then
(i) all its tests in $T_1, T_2, \ldots, T_j$ are consistent,
(ii) there is exactly one test of x_i of rank 1, say t_1,
(iii) there is exactly one test of x_i of rank 2, say t_2.
(iv) If $j_1 \leq j_2$ are such that t_1 is contained in $\alpha_1 \in T_{j_1}$ and t_2 is contained in $\alpha_2 \in T_{j_2}$, then $|T_{j_1}| = |T_{j_2}| = 1$ and t_2 is the last test of α_2.

Note that if Requirement 1 is satisfied, there may be inconsistent tests of some variable in $T_1, T_2, \ldots, T_j$, if all have rank 1. In particular, all these tests have to be contained in the same set T_{j_1} for some $j_1 = 1, 2, \ldots, j$.

Requirement 2 (i) If T_i is assigned type A, then it contains no test of rank 2.
(ii) If T_i is assigned type B, then it contains no test of rank 2 and, moreover, it contains exactly one path.
(iii) If T_i is assigned type C, then it contains exactly one path, this path contains exactly one test of rank 2 and this test is the last test of the path.

The procedure of creating sets T_j and the assignment of types will be such that in each step of the process, Requirements 1 and 2 will be satisfied.

Let a sequence $T_1, T_2, \ldots, T_{j-1}$ satisfying Requirements 1 and 2 be given. Note that, at the beginning of the process, i.e. if $j = 1$, the sequence of sets is empty and, hence, it satisfies both Requirements 1 and 2. In order to describe the procedure of creating the set T_j, choose any path α starting at v_{j-1} and calculate the ranks of tests in α according to the sequence $T_1, T_2, \ldots, T_{j-1}, \{\alpha\}$ of j sets.

Definition 5.2 A path α starting in v_{j-1} is called *good*, if there is no test of rank 2 in α.

Let α be a good path starting at v_{j-1}. There are two kinds of tests in α, tests of rank 1 and tests of rank at least 3. Note that if some variable has a test of rank 1 in α, then there is no other test of the same variable in α and also no test of the variable in $T_1, T_2, \ldots, T_{j-1}$. Moreover, if some variable has a test of rank at least 3, then also the first test of this variable in α has rank at least 3. Hence, the variable is repeated already in $T_1, T_2, \ldots, T_{j-1}$ and then all occurrences of this variable in $T_1, T_2, \ldots, T_{j-1}$ are consistent. Hence, every test of such a variable in α is either consistent with all its preceeding tests in $T_1, T_2, \ldots, T_{j-1}$ or with none of them.

Definition 5.3 (i) A good path α is called *consistent*, if all tests in α of any variable that is repeated in $T_1, T_2, \ldots, T_{j-1}$ are consistent with all the tests of the same variable in $T_1, T_2, \ldots, T_{j-1}$.
(ii) A good path is called *maximal* good path, if it leads to a sink or to a node labeled by a variable x_i, such that adding a test of x_i to the path creates a test of rank 2.

If a consistent good path is not maximal, then it leads to a node, such that the variable x_i tested in it is either repeated in $T_1, T_2, \ldots, T_{j-1}$ or has not test there. In the former case, one of the edges leaving the node forms a consistent prolongation of the path. In the latter case, both edges leaving the node lead to a consistent prolongation. Hence, every consistent good path is a prefix of a consistent maximal good path, i.e. of a consistent path that is moreover a maximal good path.

We shall distinguish the following three cases. It is easy to see that if Case 1 does not occur, then at least one of Cases 2 or 3 occurrs. If Case 2 and 3 occur simultaneously, Case 2 has higher priority. Consider all consistent maximal good paths starting at v_{j-1}.

Case 1: Every such path contains at least p tests of rank 1.
Case 2: Among these paths, there is a consistent good path α containing $< p$ tests of rank 1 and leading to a sink.
Case 3: Among these paths, there is a consistent good path α containing $< p$ tests of rank 1 and leading to a node w in which a variable x_i is tested, such that adding a test of x_i to the end of α produces a test of rank 2.

Now, we describe the procedure of creating the set T_j, the assignment of type to this set and the possible modifications in the previous sets.

In **Case 1**, let S be the set of all consistent good paths containing exactly p tests of rank 1 and such that the last test of the path has rank 1. Note that no path of S is a prefix of another. If a consistent good path reaches a node adding a test of rank 1, then a consistent good path may continue along both edges leaving the node. Hence, S consists of 2^p paths.

For every node u, let S_u be the subset of paths from S leading to the node u. Now, choose u so that the fluctuation of S_u relative to the variables in tests of rank 1 be maximal. Then, $v_j = u$, $T_j = S_u$ and its type is chosen to be A.

In **Case 2**, $T_j = \{\alpha\}$ and it will be considered of type B. In this situation, the process stops and t is set to j.

In **Case 3**, either there is some test of x_i (mentioned in Case 3 above) of rank 1 in $T_1, T_2, \ldots, T_{j-1}$ (Subcase 3a) or there is some test of x_i of rank 1 in α (Subcase 3b). Assume, Case 2 does not occurr. Then, the two subcases are handled as follows:

Subcase 3a. Let j_1 be such that x_i has a test in T_{j_1}. If the type assigned to T_{j_1} is A, we choose a path β containing x_i in T_{j_1}, change the set T_{j_1} to $\{\beta\}$ and its type is changed to B. After this change, the test of x_i in β is the unique test of x_i in $T_1, T_2, \ldots, T_{j-1}$. Among the two edges leaving w, we choose the edge consistent with the test of x_i in β. Let α' be the path consisting of α and the chosen edge. Finally, let $T_j = \{\alpha'\}$ and let its type be C.

Subcase 3b. Let α' be the path consisting of α and the edge leaving w, which is consistent with the test of x_i already contained in α. Then, let $T_j = \{\alpha'\}$ and let its type be C.

It is easy to verify that in each case, the new sequence of sets $T_1, T_2, \ldots, T_j$ satisfies Requirements 1 and 2.

According to the description of the process, some tests contained in the new set T_j may be later deleted, if Subcase 3a occurrs. Note, however, that if some test is not deleted until the end of the process, then, if its rank was 1, 2 or 3, it does not change and if it was more than 3, it is still at least 3 at the end of the process.

Assume, the process just described stopped with the sequence $T_1, T_2, \ldots, T_t$. Concatenation of any choice of $\alpha_i \in T_i$ for $i = 1, 2, \ldots, t$ forms a valid computation testing at most tp different variables and ending in a sink. By the assumptions of the theorem, setting of at most $n - r$ variables does not lead to constant subfunction. Hence, we have $tp > n - r \geq 2kp$ and so, $t \geq 2k + 1$.

The number of sets in the sequence $T_1, T_2, \ldots, T_t$ of type A, B or C will be denoted a, b and c respectively. The type B may be assigned to a set in $T_1, T_2, \ldots, T_t$ only in Case 2 and in Subcase 3a. Case 2 may occur only as the last step of the process and in Subcase 3a the new set is assigned type C. Hence, $c + 1 \geq b$.

Assume for a moment that there is no set of type A among the first $2k$ sets in the sequence. Then, $b + c = 2k$. Hence, we have $2c \geq 2k - 1$ and so $c \geq k$. Since c is the number of sets of type C, there is at least c tests of rank 2 occurring in sets T_j of size 1. Hence, these tests are repeated tests of different variables in any computation which may be created from $T_1, T_2, \ldots, T_t$. It is a contradiction, since P is a $(1, +(k-1))$-b. p.

Hence, there is a set T_j of type A among the first $2k$ sets of $T_1, T_2, \ldots, T_t$. Let T_j be the first of such sets. All the sets $T_1, T_2, \ldots, T_{j-1}$ are of type B or C and hence contain exactly one path. Let β be their concatenation. Note that β contains at most $(2k-1)p$ tests of different variables. If we set the variables occurring in β to the values required in β, we obtain a subprogram P' of P, in which v_{j-1} is the source.

Those variables tested in T_j that are not fixed by β have no occurrence in $T_{j+1}, \ldots, T_t$. Hence, if the fluctuation of T_j relative to these variables is at least r, it is possible to choose $\alpha_i \in T_i$ for all $i \geq j$ so that the path $\beta\alpha_j \ldots \alpha_t$ contains no test of at least r variables. This is a contradiction with the assumptions of the theorem. Hence, T_j has fluctuation less than r relative to the variables not fixed by β.

Let $T'_1, T'_2, \ldots, T'_{j-1}$ be the sequence $T_1, T_2, \ldots, T_{j-1}$ at the beginning of the step j. Since $T_i \subset T'_i$, we have that β is a concatenation of some paths chosen from $T'_1, T'_2, \ldots, T'_{j-1}$. In the following, we shall derive the properties of tests in paths in P' using the properties of these tests in the context from the step j, in which T_j was created. In particular, this means that the rank of tests in the paths starting in v_{j-1} is calculated according to $T'_1, T'_2, \ldots, T'_{j-1}$. Also the notions maximal and consistent good path are considered according to the sequence $T'_1, T'_2, \ldots, T'_{j-1}$. If we consider a test in

this way, we say that we consider the test in the original context. As a shorthand, the original rank refers to the rank considered in the original context.

Now, we are going to prove that P' is (p,r)-well-behaved. When created, the set T_j was assigned type A, otherwise, it cannot have type A at the end of the process. This may happen only in Case 1. Hence, the consistent maximal good paths in P starting at v_{j-1} satisfy the requirements of Case 1 in the original context.

Consider a path α' in P' from the source to a sink. Let α denotes α' considered in the original context. Let γ be the longest prefix of α that is good. It may be α itself or it may end in a node w such that adding to γ the edge from w which is contained in α would create a test of rank 2. In this latter case, adding any edge from w to γ leads to a test of rank 2. Hence, in both cases, γ is a maximal good path in P. Moreover, γ is consistent. It follows by the requirement of Case 1 that γ contains at least p tests of rank 1. All these tests are not influenced by the setting according β. This implies condition (i) of the definition of (p,r)-well-behaved b. p.

Claim. In each path in P', all of the first at most p tests have original rank 1.

In order to prove the claim, let α' be any path in P' containing at least one test of original rank 2. Let γ' be the prefix of α' ending just before the first of such tests. Let γ denotes γ' considered in the original context. It is a good path and any prolongation of γ by one edge contains a test of rank 2. Hence, γ is a consistent maximal good path in P. Thus, it contains at least p tests of rank 1. These tests are not influenced by the setting according to β. Hence, γ' contains at least p tests of original rank 1 and no test of original rank 2. The claim follows.

The claim implies that no two of the first at most p tests in any path in P' starting in the source may contain the same variable. This implies (ii) of the definition of (p,r)-well-behaved b. p. Moreover, the claim also implies that the fluctuation of any set of paths with p tests in P' is the same as the fluctuation of the set relative to tests of original rank 1. Together with the definition of sets S and S_u in the description of Case 1, this implies that in every S_u the fluctuation is at most the fluctuation of T_j, which is by some previous paragraph at most r. This implies (iii) of the definition of (p,r)-well-behaved b. p.

6 Proof of Theorem 3.4

Let us start the proof by the following. W.l.o.g., assume $I = \{1,2,\ldots,p\}$. Let $t = p/k$. Let ϕ_r for $r = 1,2,\ldots,t$ be the function ϕ guaranteed by the assumptions of the theorem for the k–tuple $k(r-1)+1, k(r-1)+2, \ldots, kr$. Hence, for every $j = 1,2,\ldots,m$ and every $r = 1,2,\ldots,t$, we have

$$\phi_r(v_j) = v_{j,k(r-1)+1} \oplus v_{j,k(r-1)+2} \oplus \ldots \oplus v_{j,rk}.$$

Denote by G_r for any $r = 1,2,\ldots,t$ the function defined for any $x \in \{0,1\}^p$ by

$$G_r(x) = x_{k(r-1)+1} \oplus x_{k(r-1)+2} \oplus \ldots \oplus x_{rk} \oplus \phi_r(x).$$

It is easy to see that for every $j = 1,2,\ldots,m$ and every $r = 1,2,\ldots,t$, we have $G_r(v_j) = 0$. It follows that m is at most the number of solutions in $\{0,1\}^p$ of the

system of equations $G_r(x) = 0$ for $r = 1, 2, \ldots, t$. We shall use the following lemma, which is proved in [13] as Lemma 7. We use exactly the proof from [13].

Lemma 6.1 (Sieling) *For each $J \subseteq \{1, 2, \ldots, t\}$, $J \neq \emptyset$, there are exactly 2^{p-1} elements x of $\{0,1\}^p$ so that*

$$\bigoplus_{r \in J} G_r(x) = 0. \tag{6.1}$$

Proof: Let $J \subset \{1, 2, \ldots, t\}$, $J \neq \emptyset$ be given. We partition the set $\{0,1\}^p$ into classes consisiting of two elements. Then, we show that for the two inputs x in each class, $\bigoplus_{r \in J} G_r(x)$ takes different values. Therefore, the number of inputs with $\bigoplus_{r \in J} G_r(x) = 0$ and $\bigoplus_{r \in J} G_r(x) = 1$ are equal. This implies the lemma.

We describe the partition by a procedure that computes for each input x the other member $\bar{x}$ of the class x belongs to. The $\oplus$-sum $\bigoplus_{r \in J} G_r(x)$ consists of the $\oplus$-sum $\bigoplus_{r \in J} \phi_r(x)$ and a $\oplus$-sum of $|J|k$ different single variables. The decision tree for each ϕ_r, $r \in J$ tests for the given input x at most $k-1$ variables. Hence, there are at most $|J|(k-1)$ variables tested by the decision trees for the given input. Therefore, some of the $|J|k$ single variables contained in the $\oplus$-sum is not tested for the given input x by any of the decision trees. Among these variables, choose the variable with the smallest index. We obtain $\bar{x}$ by negating this variable. In the decision trees, the same paths are selected for x and $\bar{x}$, since the variable differing these two inputs is not tested on any of the paths. This implies $\bar{\bar{x}} = x$ and therefore, this procedure really gives a partition of $\{0,1\}^p$ into two-element sets.

Moreover, for every $x \in \{0,1\}^p$, $\bigoplus_{r \in J} \phi_r(x) = \bigoplus_{r \in J} \phi_r(\bar{x})$, since in the decision trees, the same paths are chosen. On the other hand, the $\oplus$-sum of single variables takes different values, since the variable differing x and $\bar{x}$ is contained there. Therefore, we have $\bigoplus_{r \in J} G_r(x) \neq \bigoplus_{r \in J} G_r(\bar{x})$. $\square$

Using Lemma 6.1, we shall prove the following statement:

Lemma 6.2 *For every $J \subseteq \{1, 2, \ldots, t\}$ and every choice of $c_r \in \{0,1\}$ for all $r \in J$, the system of equations*

$$G_r(x) = c_r \quad \text{for} \quad r \in J \tag{6.2}$$

has exactly $2^{p-|J|}$ solutions.

Here, we assume that every element of $\{0,1\}^p$ satisfies the system of equations, if $J = \emptyset$. Lemma 6.2 implies Theorem 3.5, if we choose $J = \{1, 2, \ldots, t\}$ and $c_r = 0$ for all $r \in J$.

Proof: For $J = \emptyset$, the statement is trivial. Fix some nonempty J and some right hand sides (RHSs) c_r in the system (6.2) for all $r \in J$. By induction, assume that the lemma is true for every proper subset of J and any choice of the RHSs. Let $c_r[0] = c_r$ for $r \in J$. Let h be the number of $r \in J$ for which $c_r[0] = 1$. Construct a sequence $c_r[i]$ for $r \in J$ and $i = 0, 1, \ldots, h$ such that:

(i) For all $r \in J$, $c_r[h] = 0$.
(ii) For every $i = 0, 1, \ldots, h-1$, there is exactly one $r \in J$ such that $c_r[i] \neq c_r[i+1]$.

For $i = 0, 1, \ldots, h$, let a_i be the number of solutions of (6.2) with the RHSs $c_r[i]$. Let M

be the number of solutions of (6.2) with zero RHSs. First, we shall express a_i in terms of M. In particular, $a_h = M$. In order to express the other a_i, notice that for every $i = 0, 1, \ldots, h-1$, the number $a_i + a_{i+1}$ is exactly the number of solutions of a system of $|J| - 1$ equations, obtained by removing the equation r, for which $c_r[i] \neq c_r[i+1]$. By the induction hypothesis, the number of solutions of this reduced system is $2^{p-|J|+1}$. This implies that for even i, $a_{h-i} = M$ and for odd i, $a_{h-i} = 2^{p-|J|+1} - M$. Since $c_r[0]$ was an arbitrary choice of the RHSs in (6.2), we have that for any choice c_r of the RHSs, the number of solutions of (6.2) is either $2^{p-|J|+1} - M$ or M if the number of $r \in J$ for which $c_r = 1$ is odd or even respectively.

In order to determine M, we use Lemma 6.1. Namely, we express the number of solutions to (6.1) for our choice of J in terms of M and we compare the result with the value given by Lemma 6.1. An element of $\{0,1\}^p$ is a solution of equation (6.1), if and only if it is a solution of (6.2) with a RHS with an even number of ones. Clearly, two different choices of the RHSs in (6.2) lead to disjoint sets of solutions. Since we have $2^{|J|-1}$ possibilities of the RHSs with an even number of ones, the number of solutions of (6.1) is $2^{|J|-1}M$. By Lemma 6.1, this should be equal to 2^{p-1}, hence $M = 2^{p-|J|}$. It follows that for every choice of the RHSs, (6.2) has exactly $2^{p-|J|}$ solutions. $\square$

As stated above, this implies Theorem 3.5, if we choose $J = \{1, 2, \ldots, t\}$ and $c_r = 0$ for all $r \in J$.

Bibliography

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

[9] 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.

[10] K. Prachar, *Distribution of Prime Numbers* (in German), *Primzahlverteilung*, Springer-Verlag, Berlin–Göttingen–Heidelberg, 1957.

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

[12] P. Savický, S. Žák, A Large Lower bound for 1-branching programs, TR96-036, ECCC, Trier.

[13] 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*.

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.